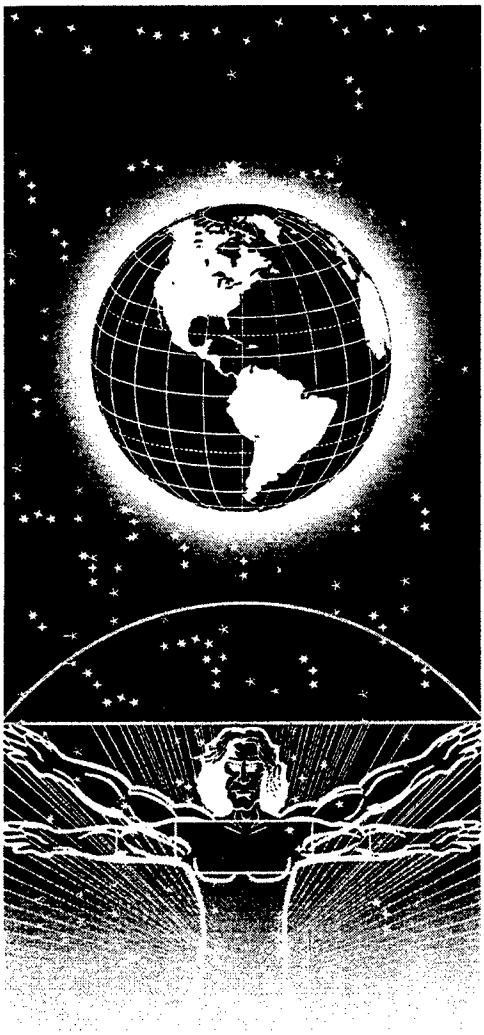
AFRL-HE-BR-TR-1998-0105



UNITED STATES AIR FORCE RESEARCH LABORATORY

A NON-PARAMETRIC ANALYSIS OF MORBIDITY/MORTALITY DATA

Daniel Mihalko

DEPARTMENT OF STATISTICS
WESTERN MICHIGAN UNIVERSITY
Kalamazoo, MI 49008

Joel E. Michalek

HUMAN EFFECTIVENESS DIRECTORATE
DIRECTED ENERGY BIOEFFECTS DIVISION
AIR FORCE HEALTH STUDIES
2606 Doolittle Road
Brooks Air Force Base TX 78235-5250

November 1998

NOTICES

This report is published in the interest of scientific and technical information exchange and does not constitute approval or disapproval of its ideas or findings.

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the US Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

The Office of Public Affairs has reviewed this technical report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

JOEL E. MICHALEK, Ph.D
Project Scientist

RICHARD L. MILLER, Ph.D
Chief, Directed Energy Bioeffects Division

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 1999	Final Report - January 1996 - September 1998

4. TITLE AND SUBTITLE
A Non-parametric Analysis of Morbidity/Mortality Data

5. FUNDING NUMBERS
PE – 65306F
PR – 2767
TA – YA
WU – 01

6. AUTHOR(S)
Daniel Mihalko
Joel E. Michalek

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Statistics
Western Michigan University
Kalamazoo, MI 49008

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory (AFMC)
Human Effectiveness Directorate
Directed Energy Bioeffects Division
Air Force Health Studies
2606 Doolittle Road
Brooks Air Force Base TX 78235-5250

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-HE-BR-TR-1998-0105

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The joint distribution of the time-to-onset of disease and the time-to-death is estimated using nonparametric methods. A wide variety of censoring patterns for both times-to-event are accommodated. The Estimation Maximization (EM) algorithm is used to obtain maximum likelihood estimates of joint probabilities. The covariance matrix for the estimates is also estimated. The methodology is demonstrated with an example derived from an epidemiological study.

14. SUBJECT TERMS
Estimation Maximization algorithm
Morbidity/Mortality

15. NUMBER OF PAGES
110

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev.2-89)
Prescribed by ANSI Std. Z39-18
298-102

CONTENTS

TABLES

TABLES (Continued)

1 Introduction

Analyses of time to disease onset and, separately, time to death in cohort studies are well established; see, for example, Lee (1980). Analyses that account for the natural progression of disease and death are lacking, however. For example, a standard device is to ignore time and score the occurrence of disease or death with or from the disease as a failure. To account for time, investigators sometimes use the first appearance of the disease, either at diagnosis or at death. Neither of these approaches explicitly distinguishes chronic from rapidly lethal diseases or the effect of the treatment on lethality. To that end, we consider the nonparametric estimation of the joint distribution of the time-to-onset of a specific disease and time-to-death in cohort studies.

Time-to-onset and time-to-death data are subject to many possible patterns of right and left censoring on the pairs of times-to-event. As pointed out by Kaplan and Meier (1958), the nonparametric approach to distribution estimation requires the discretizing of continuous data. In the case of the two dimensional time-to-event vectors considered in this paper, discretizing the data means partitioning the two dimensional space determined by time-to-onset and time-to-death into rectangles. A joint probability estimate will then be an estimate of the probability of a particular rectangle. The case in which death occurs before the onset of disease is also covered by our methods.

Let T denote the time-to-onset of the disease onset and D denote the time-to-death. These times may be measured from birth or some other point in time, such as the beginning of a period of exposure to an environmental contaminant. Our goal is to estimate the probability distribution of the pair (T,D). The basic method for nonparametric estimation of this distribution is to divide the first quadrant of the two dimensional (T,D) plane into rectangles $R_{11}, \ldots, R_{IJ}$ and estimate the probability that (T,D) will fall in R_{ij} by the sample proportion of observed (T,D) pairs that occurred in R_{ij}. Deaths without disease are included by augmenting the set of two dimensional rectangles by the set of intervals $R_{.1}, \ldots, R_{.J}$ on the positive real line, which are interpreted as intervals for death times of individuals who died without the disease. The probability estimates are the sample proportion of those individuals who died without the disease and whose death times fell into the interval $R_{.j}$, $j = 1, \ldots, J$.

Morbidity/mortality data are subject to a high degree of censoring on one or both of the members of the pair (T,D) because the disease may not be rapidly lethal, death can occur prior to the onset of disease and because many studies are analyzed before all of the subjects have died. Both T and D may be subject to left as well as right censoring. For example, the disease may have been present at the time that the subject entered the study, in which case T would be left censored.

We use the iterative Estimation Maximization (EM) algorithm of Dempster, Laird and Rubin (1977) to find the maximum liklihood estimate of the joint distribution of (T,D). At any iteration, this algorithm distributes censored observations among the rectangles that are consistent with the observed censoring pattern according to a current set of estimated probabilities. When all observations have been distributed in this way, each rectangle will have a set of "pseudo-counts" assigned to it. Pseudo-proportions are calculated using the pseudo-counts. The pseudo-proportions are used in the next iteration as a new set of estimated probabilities to redistribute the censored observations producing another set of pseudo-counts. The iterative process continues until the sequence of estimated probabilities converges. We distribute each observation separately among its potential rectangles because the final set of distributions for each observation is needed to apply the method of Louis (1982) for estimating the covariance matrix of the vector of probability estimates produced by the EM algorithm.

2 Notation

The first quadrant of two dimensional (T,D) space is partitioned by rectangles $R_{ij}, i = 1, \ldots, I$ and $j = 1, \ldots, J$, where the boundaries of the rectangles R_{ij} are defined by the points r_ℓ, $\ell = 1, \ldots, \ max(I, J)$, according to

$$R_{ij} = \{(\mathrm{t,d}):\ r_i \leq \mathrm{t} < r_{i+1},\ r_j \leq \mathrm{d} < r_{j+1}\},$$

where r_{I+1} and r_{J+1} are defined to be ∞. We also define

$$R_{.j} = \{(\mathrm{t,d}):\ \mathrm{t} > \mathrm{d} \text{ and } r_j \leq \mathrm{t}\ < r_{j+1}\},$$

which are the rectangles that account for deaths without disease. The resulting probability distribution is defined as

$$p_{ij} = P((\mathrm{T,D}) \,\epsilon\, R_{ij})\ \ , i = 1, \ldots, I,\ \ j = 1, \ldots, J,$$

and

$$p_{.j} = P((\mathrm{T,D}) \,\epsilon\, R_{.j})\ , j = 1, \ldots, J.$$

Using the same set of endpoints for both disease and death allows the determination of the potential intervals for a censored observation.

The censoring patterns can be described in the following way. For a time-to-event random variable X, denote two independent censoring variables as cXL and cXR. X is left censored and cXL is observed if X < cXL. X is right censored and cXR is observed if X > cXR. X is interval censored if cXR < X < cXL, in which case the information about X is that it lies in the interval (cXR, cXL). Thus, four censoring indicators, two for T and two for D, are required. These are defined as

iTL = 1 if T is left censored, 0 otherwise
iTR = 1 if T is right censored, 0 otherwise
iDL = 1 if D is left censored, 0 otherwise
iDR = 1 if D is right censored,0 otherwise.

3 The Discrete Likelihood

In the discrete model with no censoring, one observes a frequency for each of (I+1)J rectangles, R_{ij} and $R_{.j}$, $i = 1, \ldots, I$ and $j = 1, \ldots, J$. When censoring is present, each observation gives information about (T,D) via its censoring pattern, given by the vector of censoring indicators, (iTL, iTR, iDL, iDR), and the observed values of T and D or their censoring variables. This information indicates which of the R_{ij} or $R_{.j}$ are possible for the actual value of the vector (T,D). There are 2^4 censoring patterns for cases for which it is known that disease occurred before death. There are four censoring patterns for cases for which it is known that death occurred before disease and there are four censoring patterns for cases is which nothing is known about the time-to-onset of disease. Table 1 lists the censoring patterns and each pattern's contribution to the discrete likelihood for the cases for which something is known about the time-to-onset of disease. Table 2 lists the censoring patterns and likelihood contibutions for the cases for which nothing is known about the time-to-onset of disease. In Table 1, the expression T= r_u indicates that $r_u \leq T < r_{u+1}$. Similar notation is used for D in both Tables 1 and 2. In general, T and D are considered discrete with possible values $r_1, \ldots, r_{\max(I,J)}$.

Table 1
Censoring Patterns and Likelihood Contributions

Censoring Pattern	(iTL,iTR,iDL,iDR)	T	D	Likelihood Contribution
1	(0,0,0,0)	$T = r_u$	$D = r_s$	p_{us}
2	(1,0,0,0)	$T \leq r_u$	$D = r_s$	$\sum_{q=1}^{u} p_{qs}$
3	(0,1,0,0)	$T \geq r_u$	$D = r_s$	$\sum_{q=u}^{min(I,s)} p_{qs}$
4	(0,0,1,0)	$T = r_u$	$r_u \leq D \leq r_s$	$\sum_{\ell=u}^{s} p_{u\ell}$
5	(0,0,0,1)	$T = r_u$	$r_u \leq r_s \leq D$	$\sum_{\ell=s}^{J} p_{u\ell}$
6	(1,1,0,0)	$r_L \leq T \leq r_u$	$D = r_s$	$\sum_{q=L}^{u} p_{qs}$
7	(1,0,1,0)	$T \leq r_u$	$T \leq D \leq r_s$	$\sum_{q=1}^{u} \sum_{\ell=q}^{s} p_{q\ell}$
8	(1,0,0,1)	$T \leq r_u$	$T \leq r_s \leq D$	$\sum_{q=1}^{u} \sum_{\ell=s}^{J} p_{q\ell}$
9	(0,1,1,0)	$T \geq r_u$	$T \leq D \leq r_s$	$\sum_{q=u}^{min(I,s)} \sum_{\ell=q}^{s} p_{q\ell}$
10	(0,1,0,1)	$T \geq r_u$	$T \leq r_s \leq D$	$\sum_{q=u}^{s} \sum_{\ell=s}^{J} p_{q\ell}$
11	(0,0,1,1)	$T = r_u$	$r_u \leq r_L \leq D \leq r_s$	$\sum_{\ell=L}^{s} p_{u\ell}$
12	(1,1,1,0)	$r_L \leq T \leq r_u$	$T \leq D \leq r_s$	$\sum_{q=L}^{u} \sum_{\ell=q}^{s} p_{q\ell}$
13	(1,1,0,1)	$r_L \leq T \leq r_u$	$r_u \leq r_s \leq D$	$\sum_{q=L}^{u} \sum_{\ell=s}^{J} p_{q\ell}$

Table 1 (Continued)

Censoring Pattern	(iTL,iTR,iDL,iDR)	T	D	Likelihood Contribution
14	(1,0,1,1)	$T \leq r_u$	$r_u \leq r_L \leq D \leq r_s$	$\sum_{q=1}^{u}\sum_{\ell=L}^{s} p_{q\ell}$
15	(0,1,1,1)	$r_u \leq T$	$\max(T,r_L) \leq D \leq r_s$	$\sum_{q=u}^{s}\sum_{\ell=\max(q,L)}^{s} p_{q\ell}$
16	(1,1,1,1)	$r_L \leq T \leq r_u$	$r_u \leq r_t \leq D \leq r_s$	$\sum_{q=L}^{u}\sum_{\ell=t}^{s} p_{q\ell}$
17	(.,.,0,0)	no T	$D = r_s$	$p_{.s}$
18	(.,.,1,0)	no T	$D \leq r_s$	$\sum_{\ell=1}^{s} p_{.\ell}$
19	(.,.,0,1)	no T	$r_s \leq D$	$\sum_{\ell=s}^{J} p_{.\ell} + \sum_{q=s}^{I}\sum_{\ell=q}^{J} p_{q\ell}$
20	(.,.,1,1)	no T	$r_L \leq D \leq r_s$	$\sum_{\ell=L}^{s} p_{.\ell} + \sum_{q=L}^{s}\sum_{\ell=q}^{s} p_{q\ell}$

In censoring patterns 17 through 20, the disease is known not to have occurred before death or before the observed censoring time or times for death. In pattern 17, the death time is uncensored and it is known that the subject died without the disease. In pattern 18, death is left censored so that the only information about time-to-onset of the disease is that it did not occur before the left censoring value of time-to-death. In pattern 19, time-to-death is right censored so that it is only known that the onset of disease did not occur before this censoring point, but may have occurred between the right censoring time for time-to-death and the actual time-to-death. In pattern 20, the time-to-death is interval censored and it is assumed that the time-to-onset did not occur before the left endpoint of the interval censoring time-to-death.

Table 1 lists censoring patterns, for which at least some information is known about both time-to-onset of disease and time-to-death. It is possible that only time-to-death is known and that no information regarding disease history is available. In this case, technically, time-to-onset of disease is right censored at time zero. There are four possible patterns for this situation, corresponding to the four possible censoring patterns for time-to-death. Table 2 lists these four possible censoring patterns and their likelihood contributions.

Table 2
Censoring Patterns and Likelihood Contributions
For Subjects with Time-to-Death Information Only

Censoring Pattern	(iTL,iTR,iDL,iDR)	T	D	Likelihood Contribution
21	(0,1,0,0)	T≥ 0	D$= r_s$	$\sum_{q=0}^{min(I,s)} p_{qs} + p_{.s}$
22	(0,1,1,0)	T≥ 0	D$\leq r_s$	$\sum_{q=0}^{min(I,s)} \sum_{\ell=q}^{s} p_{q\ell} + \sum_{\ell=1}^{s} p_{.\ell}$
23	(0,1,0,1)	T≥ 0	$r_s \leq$ D	$\sum_{q=0}^{I} \sum_{\ell=\max(q,s)}^{J} p_{q\ell} + \sum_{q=s}^{I} \sum_{\ell=q}^{J} p_{q\ell}$
24	(0,1,1,1)	T≥ 0	$r_L \leq$ D$\leq r_s$	$\sum_{q=0}^{s} \sum_{\ell=\max(q,L)}^{s} p_{q\ell} + \sum_{\ell=L}^{s} p_{.\ell}$

4 The EM Algorithm

Each likelihood contribution listed in Tables 1 and 2 is the sum of all of the probabilities of rectangles that are consistent with the censoring pattern. The EM algorithm distributes the censored observations as pseudo-counts among the rectangles that are consistent with the censoring pattern. The distribution of censored observations is defined by the conditional probability that the actual value of (T,D) would occur in the rectangle given the censoring pattern. This conditional probability is the ratio of the unconditional probability of (T,D) occurring in the rectangle divided by the sum of probabilities for all rectangles consistent with the censoring pattern. This ratio is simply the unconditional rectangle probability, p_{ij} or $p_{.j}$, divided by the likelihood contribution as listed in Tables 1 and 2. For example, an observation with censoring pattern 1 (Table 1) is completely uncensored and thus contributes 0 to all rectangles except for R_{us}, which is allocated a 1. Letting P_i indicate the likelihood contribution of an observation of censoring pattern i, an observation with censoring pattern 2 contributes p_{ij}/P_2 to rectangles R_{ij}, where $i = 1, \ldots, u$ and $j = s$ and allocates 0 to all other rectangles. Each censoring pattern defines a set of possible rectangles in which the pair (T,D) could have actually occurred. The EM algorithm begins with a set of rectangle probabilities, p^0_{ij} and $p^0_{.j}, i = 1, \ldots, I$ and $j = 1, \ldots, J$. The sum of all data allocations for each rectangle form the first set of pseudo-counts, c^1_{ij} and $c^1_{.j}, i = 1, \ldots, I$ and $j = 1, \ldots, J$, for the rectangles. In the next iteration, a new set of rectangle probabilities, p^1_{ij} and $p^1_{.j}, i = 1, \ldots, I$ and $j =$

$1, \ldots, J$, are estimated by the sample proportions of the pseudo-counts. That is, according to

$$p^1_{ij} = c^1_{ij}/n, \quad i = 1, \ldots, I \text{ and } j = 1, \ldots, J,$$

$$p^1_{.j} = c^1_{.j}/n, \quad j = 1, \ldots, J,$$

where n is the total sample size. In the following iteration, the new set of probabilities is used to reallocate each observation to form another set of pseudo-counts. The process continues until the rectangle probabilities have converged.

5 The Information Matrix

Louis (1982) presented a method for estimating the information matrix of a vector of parameter estimates obtained using the EM algorithm. The application of that method to the rectangle probability estimates is straightforward. Each observation can be represented as a vector of indicator functions

$$(x_{.1}, \ldots, x_{.J}, x_{11}, x_{12}, \ldots, x_{IJ}),$$

where, in the complete data case, one and only one of the components is 1 and the others are 0. The probability distribution for each observation can be written as a vector of multinomial probabilities

$$(p_{.1}, \ldots, p_{.J}, p_{11}, p_{12}, \ldots, p_{IJ}).$$

The lower triangular portion of the submatrix formed by eliminating the death-without-disease row must have zero probabilities, because these rectangles represent the impossibility of disease occurring after death. Eliminating these impossible rectangles from the observation and parameter vectors and renumbering the remaining rectangles across rows and down columns (the method that SAS^{TM} uses for numbering two dimensional arrays), we can write the observation and parameter vectors as,

$$\mathbf{x} = (x_1, \ldots, x_K)$$

and

$$\mathbf{p} = (p_1, \ldots, p_K),$$

where $K = (I+1)J - (I-1)I/2$, the number of possible rectangles. Letting

$$f(\mathbf{x}|\mathbf{p}) = f(\mathbf{x}|p_1, \ldots, p_{K-1}),$$

the probability distribution function for the observation vector can be written as,

$$f(\mathbf{x}|\mathbf{p}) = \sum_{k=1}^{K-1} x_k p_k - x_K(1 - \sum_{k=1}^{K-1} p_k).$$

Following Louis (1982), let

$$S(\mathbf{x}, \mathbf{p}) = (\frac{\partial}{\partial p_1} f(\mathbf{x}|\mathbf{p}), \ldots, \frac{\partial}{\partial p_{K-1}} f(\mathbf{x}|\mathbf{p}))^T.$$

For our multinomial case the value of $S(\mathbf{x}, \mathbf{p})$ is

$$S(\mathbf{x}, \mathbf{p}) = (\frac{x_1}{p_1} - \frac{x_K}{p_K}, \ldots, \frac{x_{K-1}}{p_{K-1}} - \frac{x_K}{p_K})^T,$$

where $p_K = 1 - \sum_{k=1}^{K-1} p_k$.

An estimate of the information matrix for the estimates of $(p_1, \ldots, p_{K-1})$ obtained by the EM algorithm is,

$$\mathrm{I}_{K-1} = \sum_{r=1}^{n} S^T(\mathbf{x_r}, \mathbf{p}) S(\mathbf{x_r}, \mathbf{p}),$$

where $\mathbf{x_r}$ is the rth pseudo-data observation from the results of the converged EM algorithm.

6 Epidemiological Measures

Given a large sample, the vector of estimates

$$\hat{\mathbf{p}} = (\hat{p}_1, \ldots, \hat{p}_{K-1})$$

is approximately multivariate normally distributed with covariance matrix I_{K-1}^{-1}. Therefore, the delta method (Bishop, Feinberg and Holland (1975), page 486) can be applied to obtain the asymptotic distribution of any differentiable function of $\hat{\mathbf{p}}$.

For convenience, we compute the singular covariance matrix of the entire vector of estimates $(\hat{p}_1, \ldots, \hat{p}_K)$. This covariance matrix, Σ_K, is I_{K-1}^{-1} augmented by the following Kth row and Kth column.

$$\text{Kth row} = (-\sum_{k=1}^{K-1} \sigma_{k1}, \ldots, -\sum_{k=1}^{K-1} \sigma_{k1}, \sum_{k=1}^{K-1} \sum_{s=1}^{K-1} \sigma_{ks}),$$

where σ_{ks} is the (k, s) element of I_{K-1}^{-1}. The Kth column is the transpose of the Kth row.

The morbidity/mortality experience of two groups can be compared by testing the equality of their joint probability distributions, assuming the same set of rectangle boundaries for both groups. Let the vectors of estimates be given by $\mathbf{p_1}$ and $\mathbf{p_2}$ for each group and the covariance matrices be given by $\Sigma_{1,K}$ and $\Sigma_{2,K}$. Then the statistic

$$(\mathbf{p_1} - \mathbf{p_2})(\Sigma_{1,K} + \Sigma_{2,K})^{-}(\mathbf{p_1} - \mathbf{p_2})^T,$$

where the exponent "-" indicates the generalized inverse, has approximately a chi-square distribution with $K - 1$ degrees of freedom under the null hypothesis that the two joint distributions are the same.

Relative Risk:

Given two groups, say exposed (group 1) and controls (group 2), four relative risk measures can be estimated. The relative risk of dying with disease in interval j after contracting the disease in interval i is defined by

$$\mathrm{RR}_{ij} = p_{1,ij}/p_{2,ij}, \quad i = 1, \ldots, \mathrm{I}, \quad j = 1, \ldots, \mathrm{J},$$

where $p_{1,ij}$ and $p_{2,ij}$ are the probabilities for the rectangle R_{ij} for group 1 and group 2. The relative risk of dying without disease in interval j is defined by

$$\mathrm{RR}_{.j} = p_{1,.j}/p_{2,.j}, \quad j = 1, \ldots, \mathrm{J},$$

where $p_{1,j}$ and $p_{2,j}$ are the jth probabilities in the vector for groups 1 and 2, respectively. The relative risk of dying with disease in interval j is

$$\mathrm{RR}_j = (\sum_{i=1}^{j} p_{1,ij})/(\sum_{i=1}^{j} p_{2,ij}), \quad j = 1, \ldots, J,$$

and the overall relative risk of dying with disease is

$$\mathrm{RR} = (\sum_{i=1}^{I}\sum_{j=1}^{J} p_{1,ij})/(\sum_{i=1}^{I}\sum_{j=1}^{J} p_{2,ij}).$$

Estimates of these relative risks are ratios of single probability estimates or sums of probability estimates. Assuming the samples from groups 1 and 2 are independent, the numerators and denominators are independent. The variance of a sum of estimators is the sum of all the entries in the covariance matrix corresponding to the summands. These variances can be calculated given the covariance matrix for the joint probability estimates from both groups. Writing the relative risk (RR) as $RR = A/B$ and letting σ_1 and σ_2 be the variances of A and B and σ_{12} the covariance of A and B, the variance of the relative risk is

$$\sigma_{RR} = \frac{1}{A^2}(\sigma_1 - 2\sigma_{12}\frac{A}{B} + \sigma_2(\frac{A}{B})^2).$$

Lethality:

Lethality is defined as the ratio of the probability of dying with the disease to the probability of dying without the disease. The lethality specific to a time-to-death interval j is the ratio of the probability of dying in interval j with disease to the probability in interval j without the disease. With respect to the matrix of joint probabilities the lethality for interval j is the sum of all probabilities for the jth column from rows 1 to I divided by the death-without-disease probability in column j. The formula for the lethality in column j is

$$\mathrm{L}_j = (\sum_{i=1}^{I} p_{ij})/p_{.j}, \quad j = 1, \ldots, \mathrm{J}.$$

The covariance matrix for the vector of lethalities, $\mathbf{L}^T = (\mathrm{L}_1, \ldots, \mathrm{L}_J)$, is computed using the gradient matrix, grad$\mathbf{L}$, defined as the J by K matrix whose (r,s) element is $\frac{\partial}{\partial p_s} L_r$.

The delta method then implies that $\mathbf{L}$ is approximately multivariate normally distributed with covariance matrix equal to

$$\Sigma_L = (\mathrm{grad}\mathbf{L})\Sigma_K(\mathrm{grad}\mathbf{L})^T.$$

The lethalities of the two groups can be compared by using the chi-square statistic

$$(\mathrm{grad}\mathbf{L_1} - \mathrm{grad}\mathbf{L_2})(\Sigma_{L_1} + \Sigma_{L_2})^{-}(\mathrm{grad}\mathbf{L_1} - \mathrm{grad}\mathbf{L_2})^T,$$

where the degrees of freedom are equal to the rank of $\Sigma_{L_1} + \Sigma_{L_2}$.

7 Example

This example compares the morbidity/mortality experience of two groups. One group (group E) was exposed to an environmental contaminant, while the other group (group C) was not exposed. Under the null hypothesis, both groups are random samples from the same population. Deaths or disease that had not occurred by the time the data was collected were right censored at the calendar time of the collection. Table 3 shows the distributions of the censoring patterns for both groups.

Table 3
Censoring Pattern Distributions by Group

	Group E		Group C	
Pattern	Frequency	Percent	Frequency	Percent
1	39	3.2	51	0.3
5	461	37.3	575	3.0
8	1	0.1	1	0.0
17	36	2.9	50	0.3
19	632	51.2	1016	5.3
21	42	3.4	1572	8.2
23	24	1.9	15791	82.9

Five-year intervals were chosen. Since the first three five-year intervals were sparse, the first interval was taken to be [0,15). Subsequent intervals were defined in increments of five years and the final interval was [35,+). The completely uncensored data (censoring patterns 1 and 17) are given in Table 4. There were no uncensored observations with time-to-death greater than 30 in group E. Because the rectangle boundaries must be the same for both groups, the algorithm reduced the the time intervals for group C to end with [25,+).

Table 4
Frequencies for Uncensored Data by Group

Group E		Time-to-Death				
	Time-to-Onset	[0,15)	[15,20)	[20,25)	[25,+)	Total
	death w/o disease	2	13	16	5	36
	[0,15)	2	7	4	0	13
	[15,20)	0	8	7	0	15
	[20,25)	0	0	6	2	8
	[25,+)	0	0	0	3	3
	Total	4	28	33	10	75
Group C		Time-to-Death				
	Time-to-Onset	[0,15)	[15,20)	[20,25)	[25,+)	Total
	death w/o disease	3	14	25	8	50
	[0,15)	1	4	2	2	9
	[15,20)	0	6	7	4	17
	[20,25)	0	0	10	7	17
	[25,+)	0	0	0	8	8
	Total	4	24	44	29	101

In group E, the rectangles with time-to-onset in [0,15) and [15,20) and time-to-death in [25,+) have 0 counts. The algorithm replaced these with the sum of the counts in the adjacent rectangles divided by the total number of uncensored patterns. For example, the rectangle with time-to-onset interval [0,15) and time-to-death interval [25,+) is given a count of (16+5+4+7)/75 (e.g. 32/75) to begin the EM algorithm.

The final probability estimates for group E and their standard deviations resulting from the application of the EM algorithm and the inversion of the information matrix are given in Table 5. The corresponding results for group C are given in Table 6.

Table 5
Joint Probabilities for Group E

		Time-to-Death			
Time-to-Onset		[0,15)	[15,20)	[20,25)	[25,+)
death w/o disease	Relative Risk	0.016599	0.012030	0.013348	0.157371
	Std Deviation	0.008682	0.003220	0.003292	0.047070
[0,15)	Relative Risk	0.016599	0.006478	0.003337	0.126720
	Std Deviation	0.008682	0.002403	0.001663	0.009579
[15,20)	Relative Risk	0	0.007403	0.005840	0.149910
	Std Deviation	0	0.002562	0.002194	0.010280
[20,25)	Relative Risk	0	0	0.005006	0.073713
	Std Deviation	0	0	0.002033	0.007522
[25,+)	Relative Risk	0	0	0	0.405646
	Std Deviation	0	0	0	0.046796

Table 6
Joint Probabilities for Group C

		Time-to-Death			
Time-to-Onset		[0,15)	[15,20)	[20,25)	[25,+)
death w/o disease	Relative Risk	0.025804	0.010770	0.012631	0.269909
	Std Deviation	0.004392	0.001600	0.001999	0.034742
[0,15)	Relative Risk	0.009824	0.002760	0.001889	0.076160
	Std Deviation	0.004215	0.001139	0.000826	0.005621
[15,20)	Relative Risk	0	0.003523	0.003456	0.128231
	Std Deviation	0	0.001372	0.001426	0.008000
[20,25)	Relative Risk	0	0	0.006163	0.158036
	Std Deviation	0	0	0.001594	0.011426
[25,+)	Relative Risk	0	0	0	0.290843
	Std Deviation	0	0	0	0.032916

Inverting the information matrices for each group and computing the chi-square statistic for the difference of the joint probability distributions yields,

$$(\mathbf{p}_R - \mathbf{p}_C)(\Sigma_R + \Sigma_C)^{-}(\mathbf{p}_R - \mathbf{p}_C)^T = 66.358,$$

which has 13 degrees of freedom (because there are 14 possible rectangles) and a p-value less than 0.0001. Tables 7 and 8 give the relative risks of a morbidity/mortality event in rectangle R_{ij}. The overall relative risk of dying with disease is 1.1759 with standard deviation 0.092775.

Table 7
Relative Risks

		Time-to-Death			
Time-to-Onset		[0,15)	[15,20)	[20,25)	[25,+)
death w/o disease	Relative Risk	0.646281	1.116966	1.056730	0.583053
	Std Deviation	0.353840	0.341972	0.309622	0.189854
[0,15)	Relative Risk	1.689636	2.347056	1.766675	1.663865
	Std Deviation	1.143107	1.302293	1.171078	0.178857
[15,20)	Relative Risk		2.101399	1.689729	1.169062
	Std Deviation		1.094568	0.942833	0.108385
[20,25)	Relative Risk			0.812170	0.466431
	Std Deviation			0.391059	0.058335
[25,+)	Relative Risk				1.394724
	Std Deviation				0.225397

Table 8
Relative Risks by Time-to-Death Interval

Time-to-Death	Relative Risk	Standard Deviation
[0,15)	1.689636	1.143106
[15,20)	2.209311	0.775555
[20,25)	1.232381	0.361280
[25,+)	1.157267	0.094997

Table 9 gives the lethality estimates by time-to-death intervals and their covariances.

Table 9
Lethality Estimates and Covariances

Group E			Covariance Time-to-Death			
	Time-to-Death	Lethality	[0,15)	[15,20)	[20,25)	[25,+)
	[0,15)	1.00000	1.00000	0.00000	-0.00000	-0.00000
	[15,20)	1.15385	-0.00000	0.19117	-0.00000	0.00000
	[20,25)	0.06250	-0.00000	-0.00000	0.13696	0.00000
	[25,+)	4.80386	-0.00000	0.00000	0.00000	3.00545
Group C			Covariance Time-to-Death			
	Time-to-Death	Lethality	[0,15)	[15,20)	[20,25)	[25,+)
	[0,15)	0.38072	0.05104	0.00000	-0.00000	-0.00000
	[15,20)	0.58335	0.00003	0.04927	-0.00000	0.00000
	[20,25)	0.91106	-0.00000	0.00000	0.08222	-0.00000
	[25,+)	2.42034	-0.00004	0.00002	-0.00002	0.19371

The chi-square test statistic with 4 degrees of freedom for testing the equality of the two vectors of lethalities given in Table 9 was 3.5989 with p-value 0.4630.

References

Bishop YMM, Feinberg SE and Holland PW (1975). Discrete Multivariate Analysis: Theory and Practice. The MIT Press. Cambridge.

Dempster AP, Laird NM and Rubin DB (1977). Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39, 1-22.

Kaplan EL and Meier P (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53, 457-481.

Lee ET (1980). Statistical Methods for Survival Data Analysis. Belmont: Lifetime Learning Publications.

Louis TA (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 44, 226-233.

Appendix 1: Manual for Implementing the SAS^{TM} Macros for Non-parametric Estimation of the Joint Probability Distribution of Time-to-Onset of Disease and Time-to-Death

Introduction

This manual presents a set of SAS^{TM} macros which implement the methods for morbidity/mortality analysis described in *A Non-Parametric Analysis of Morbidity/Mortality Data* by Mihalko and Michalek (1998). There are two main macros named mortmorb and mm2. All of the macros discussed here appear in Appendix 2.

The mortmorb macro accepts a sample of disease onset times and times-to-death, each of which may be right, left, or interval censored. Using instructions from the user, the macro formulates a set of rectangles in the two dimensional time-to-onset of disease and time-to-death space by using the uncensored points, including those with a known time-to-death and disease known not to have occurred before death. Each observation is classified according to its censoring pattern. The censoring patterns accommodated are those described in Tables 1 and 2 of Mihalko and Michalek (1998). Using these classifications and the formulated rectangles, the Estimation Maximization (EM) algorithm is used to distribute each observation into its possible set of rectangles. The final set of pseudodata is used to compute the maximum likelihood estimates of the probabilities of the rectangles. The algorithm of Louis (1982) is used to obtain an estimate of the Fisher information matrix for the probability estimates. The mm2 macro works in the same way as the mortmorb macro except that it accepts a prescribed set of rectangles.

This manual also shows how to use the mm2 macro to obtain the joint probability estimates for two independant samples using the same set of rectangles. Section 2 describes the format of the input data. Section 3 specifies the arguments for the main macros, mortmorb and mm2, and gives an example. Section 4 details the components of the two macros. Section 5 shows how to use the results of the macros to test for a difference between the joint distributions of two samples. Appendix 2 gives the code for producing estimates of the joint distributions used in the example of Mihalko and Michalek (1998).

Input Data Format

The input data must contain five variables each for time-to-onset of disease and time-to-death. These five variables describe the censoring pattern for

each event and give the times associated with the censoring pattern. The names and definitions of these required variables for time-to-onset of disease are:

obsT:

ObsT is the actual observed time-to-onset of disease. If time-to-onset of disease is censored on one side then obsT is the censored value. If time-to-onset is interval censored, then obsT = '.' .

obsTL:

ObsTL is equal to '.' if time-to-onset is not censored on the left and right, i.e. interval censored. Otherwise, obsTL is the left endpoint of the censoring interval.

obsTR:

ObsTR is equal to '.' if time-to-onset is not censored on the left and right. Otherwise, obsTR is the right endpoint of the censoring interval.

iTL:

ITL is the indicator of left censorship for time-to-onset of disease. ITL = 1 if time-to-onset of disease is left censored or interval censored and 0 otherwise.

iTR:

ITR is the indicator of right censorship for time-to-onset of disease. ITR = 1 if time-to-onset of disease is right censored or interval censored and 0 otherwise.

Death known to have occurred before disease:

If death is known to have occurred before disease then obsT, obsTL and obsTR should all be recorded as '.', missing, and iTL=0 and iTR=1. The reason for this coding is that, technically, disease has been right censored by death. All of the obsT, obsTL and obsTR variables are set to missing so that the macro can distinguish between the case that time-to-onset did not occur before death and the case that nothing is known about time-to-onset of disease.

If time-to-onset of disease is right censored then iTL =0, iTR=1, obsTL=obsTR = '.', missing, and obsT is the time of right censorship. This indicates that time-to-onset of disease is known to be larger than the value in the variable obsT or that death occurred before time-to-onset of disease.

The variables for time-to-death correspond to those for time-to-onset of disease and are listed below.

obsD:

ObsD is the actual observed time-to-death. If time-to-death is censored on one side then obsD is the censored value. If time-to-death is interval censored, then obsD = '.', missing.

obsDL:

ObsDL is equal to '.' if time-to-death is not censored on the left and right. Otherwise, obsDL is the left endpoint of the censoring interval.

obsDR:

ObsDR is equal to '.' if time-to-death is not censored on the left and right. Otherwise, obsDR is the right endpoint of the censoring interval.

iDL:

IDL is the indicator of left censorship for time-to-death. IDL = 1 if time-to-death is left censored or interval censored and 0 otherwise.

iDR:

IDR is the indicator of right censorship for time-to-death. IDR = 1 if time-to-death is right censored or interval censored and 0 otherwise.

The Macros Mortmorb and MM2

Two 'driver' macros, named mortmorb and mm2, implement the methods of Mihalko and Michalek (1998). Both drivers produce maximum likelihood estimates of a discretized version of the joint distribution of time-to-onset of disease and time-to-death and the associated Fisher information matrix. The driver named mortmorb determines the rectangles to be used for the joint distribution using arguments supplied by the user. The macro named mm2 accepts a set of rectangles chosen by the user.

The arguments for mortmorb are:

datafile:

The argument datafile is the name of a SAS^{TM} data set containing the ten variables described in Section 2. The datafile may contain other variables, such as an identification number for the observation and any potential explanatory variables. An output data set is produced that contains all of the information in datafile plus the results of the EM algorithm for each observation.

outprobs:

The argument outprobs should be a SAS^{TM} data set name, permanent or temporary. After completion of the macro, the data set outprobs will contain one observation with variables named newrows, newcols and a two dimensional array newp(newrows, newcols). The variable newrows will contain the number of intervals into which time-to-onset of disease has been divided for the joint probability distribution. The count in newrows will also include a category of missing to account for observations for which it is known that death occurred before onset of disease, if there were any such observations with uncensored time-to-death. The variable newcols will contain the number of intervals into which the time-to-death range is partitioned for the joint distribution. The array newp(newrows,newcols) will contain the maximum likelihood estimates of the joint probabilities for the array of rectangles defined by the cross tabulation of the time-to-onset of disease rows and time-to-death columns. The lower triangular part of the submatrix composed of this matrix without the row corresponding to death before disease (if such a row is included) will have zero values for the probabilities because disease cannot be observed after death. The total number of probabilities, zeros included, is the product of newrows and newcols. Letting this product be called total, the array newp will be comprised of variable newp1 through newptotal.

pseudo:

The argument pseudo should be the name of a SAS^{TM} data set which will contain all of the original data from the input data set named in the argument datafile and an array, ps(newrows,newcols), which contains the EM algorithm distribution for each observation in the newrows by newcols matrix of onset-of-disease by time-of-death rectangles. Recall that newrows and newcols are variables in the outprobs data set.

intobs:

The argument intobs determines the minimum number of uncensored observations that the user requires in each interval of the partitions of the time-to-onset space and the time-to-death space. The algorithm for determining the original partitions cuts the two time-to-event ranges into the maximum number of intervals that allow each interval to contain at least intobs uncensored times. The interval endpoints for both time-to-onset and time-to-death are then combined to form a common set of intervals which are used to define the original set of time-to-onset of disease by time-to-death rectangles. The smaller the number of intobs, the larger the number of rectangles with which the algorithm begins. The number of rectangles will be reduced later by a macro named revzero if the EM algorithm fails to put a positive count in any

rectangle for which it is possible to have an uncensored observation. This process is described in the discussion of the revzero macro.

tol:

The argument tol is the measure of convergence desired for the EM algorithm. The EM algorithm iteratively estimates the maximum likelihood estimator. After each iteration of the EM algorithm, the sum of squared differences between the current and past estimates is computed and compared to tol. The EM algorithm stops when the sum of squared differences is less than tol.

its:

The argument its is the maximum number of iterations of the EM algorithm that the user will allow. Although theory implies that the EM algorithm converges uniformly to the maximum likelihood estimate, it does not say how fast it converges. In simulations using the mortmorb macro convergence was obtained by 20 iterations using a tolerance of 0.0001. If convergence does not occur within 100 iterations it may be a good idea to increase the value of intobs and start again. The EM algorithm will continue its iterations until the sum of squared differences described in the previous paragraph is less than the value in the argument tol or until it completes the number of iterations given in the argument its. If the algorithm stops without converging, the log window will contain a message saying that convergence was not obtained in its iterations and give the values of the sum of squared errors for the last iteration of the algorithm.

libdrive:

The argument libdrive gives a computer address which the macro will use to define the library in which the formats for the partitions of the time-to-onset of disease and the time-to-death will be stored. The macro produces a SAS^{TM} format for categorizing the two times-to-events. These formats will be stored as ?dis for the time-to-onset of disease and ?deth for the time-to-death, where ? is the value in the argument pref, which stands for prefix. An example of a value for libdrive is $c:\backslash windows\backslash desktop$. Notice that no quotation marks are needed.

inform:

The argument inform is a SAS^{TM} data set name chosen by the user to contain the estimated information matrix for the estimates of the joint probability distribution. This information matrix is in a two dimensional array named im. This array will correspond to the vector of probabilities ordered across rows and down columns of the matrix of rectangles, leaving out the impossible rectangles. The last probability is not included in this

vector, because it is equal to one minus the sum of the other probabilities. Hence, if there are 14 possible rectangles, as in the example of Mihalko and Michalek (1998), the information matrix will have dimensions 13 by 13.

pref:

The argument pref is the prefix that the user chooses to be put on the formats for time-to-onset and time-to-death. Pref can have at most 4 characters. The formats for time-to-onset and time-to-death will be named prefdis and prefdeth, respectively, where pref is replaced by the character string value chosen for the argument pref. In addition, two other formats, named prefrow and prefcolumn, are saved in the libref argument address. These formats replace the row and column numbers with the appropriate row and column intervals which define the matrix of rectangles.

The arguments for mm2:

The difference between the mortmorb macro and the mm2 macro is that the mm2 macro accepts a format for partitioning the times-to-onset of disease and the times-to-death. These formats are used to define the original set of time-to-onset by time-to-death rectangles. The mm2 macro then produces the same output as the mortmorb macro. In a two sample study, mortmorb could be used with the smaller sample to determine a set of rectangles and formats. These rectangles could be then used in mm2 with the second sample, so that the pseudodata and the joint probabilities for both samples are defined for the same set of rectangles.

datafile, outprobs, pseudo, tol, its, libdrive, inform, pref:

These arguments are the same as those of the same name in the mortmorb macro.

disfmt and dethfmt:

The arguments disfmt and dethfmt are the names of the formats to be used for the time-to-onset and time-to-death variables, respectively. These formats must be stored in the address given in the argument libdrive.

intimes:

The argument intimes is the name of a SAS^{TM} data set containing variables t1 through t(number of times), where the values of the t's are the end points of the intervals defined in disfmt and dethfmt.

Note on disfmt, dethfmt and intimes:

These arguments define the original set of rectangles that are used to compute the starting probabilities for the EM algorithm. Generally they will be the output formats and times from a previous run of the mortmorb macro

or the mm2 macro. However, the only requirement is that the times be the points that define the intervals formatted in disfmt and dethfmt. Not all of the times need to be used in the format for disfmt, but all the times must be used in the dethfmt. If disfmt defines a death-before-disease category and K intervals, while dethfmt defines J intervals, then the first J−1 intervals of the dethfmt must be the first J−1 intervals defined by disfmt. The Jth interval defined by disfmt must be the union of the last K−J intervals defined by dethfmt.

The Macro Components

The main macro drivers, mortmorb and mm2, share all but one macro component. We describe each macro in the order that it appears in mortmorb and then discuss mm2 by highlighting the small differences between the two drivers.

eminvals(infile,outfile,timesout,intobs,libdrive):

The macro eminvals forms the original rectangles in the time-to-onset of disease by time-to-death sample space. The input data set is named in the argument infile, which must have the form described above for the mortmorb macro argument datafile. The macro removes the uncensored observations, including those with death is known to have occurred before the onset of disease, and orders the resulting times-to-onset of disease and times-to-death separately. For each set of times to disease onset and death, the macro chooses the points in the ordering that have intobs (as defined in the mortmorb description) times between them. The resulting two sets of times are merged into one set of distinct times. These times are used in a macro named ddformat to create formats for onset times and times-to-death as well as formats relating the row and column indices to the interval definitions. The formats all have the prefix that the user puts in the argument pref, the suffix is dis for the onset times, deth for the times-to-death, row for the row index and column for the column index. The format for onset times includes a category for missing, which is assigned to an observation that has death known to have occurred before disease. These formats are used with PROC FREQ to compute the counts of the uncensored observations in each of the intervals.

The formats are stored in the address given in the argument libdrive. The value given to the argument timesout will be the name of a data set containing one observation which contains the times comprising the endpoints of the intervals. These times are in indexed variables t1 through t(number of times). The value given to the argument outfile will be the name of the data set that

contains the output of the PROC FREQ. This output is a table of counts for the uncensored observations and the rectangles defined by the two formats.

initprob(infile,outfile):

The macro initprob receives an output SAS^{TM} data set from PROC FREQ named in the argument infile and produces a SAS^{TM} data set named by the value of outfile, which will contain a set of initial probabilities for the rectangles of the table defined by the infile. The macro determines whether or not there is a row for missing, which would correspond to death known to have occurred before disease. If so, it assigns a 1 to a global macro variable named dwod (death without disease). If there were no deaths known to have occurred before disease then dwod is assigned 0. The macro assigns zeros to the lower triangle of the matrix if dwod is 0. If dwod is 1, then the macro assigns zeros to the lower triangle of the the matrix defined by eliminating the first row. If any rectangle that can legitimately be nonzero is empty, a small amount of probability is put into that rectangle. The amount of probability put into rectangles with zero counts is the sum of the counts of adjacent rectangles divided by the total number of uncensored observations. If all adjacent rectangles have zero counts then the probability put into the rectangle is 1 divided by the number of uncensored observations. To compute the initial probabilities, the actual counts for nonzero rectangles and the 'adjusted' counts for zero rectangles are summed. Then, the rectangle probabilities are computed by dividing the count or adjusted count for each rectangle by this sum of counts or adjusted counts. These initial probabilities are put into one observation with a two dimensional array named newp. The array has &numrows number of rows and &numcols number of columns, where numrows and numcols are global macro variables containing the number of rows and number of columns of the time-to-onset of disease by time-to-death matrix of rectangles. This two dimensional matrix contains the triangle of empty rectangles. Lastly, a global macro variable named total, the total number of rectangles, is produced.

In the macro mortmorb, the PROC FREQ output SAS^{TM} data set produced by eminvals is named freqout and is the infile to initprob. The outfile produced by initprob is named initprob.

cellsgn(infile,datafile,clasfile,numrow,numcol,dwod):

The macro cellsgn augments datafile with the row and column of the matrix defined in infile. The arguments numrow, numcol are the number of rows and columns of the matrix of rectangles, while dwod is the indicator as to whether or not there is a row for death without disease. The resulting SAS^{TM} data set will be named by the value given to the argument clasfile. Infile must be a SAS^{TM} data set containing one observation with an array t(numcol) of ordered times, of the type put out in the timesout argument of

eminvals. The names of the new variables in clasfile are row, rowL and rowR indicating the row classification of obsT, obsTL and obsTR, respectively. Similarly the column classifications for obsD, obsDL and obsDR are column, columnL and columnR, respectively.

In the mortmorb macro, the data set infile is named times and is the output of timesout from the eminvals macro. The arguments numrow, numcol and dwod are the global macro variables numrows, numcols and dwod created in the macro initprob. The argument for datafile is the original morbidity/mortality data set put into the argument datafile of the mortmorb macro. The output data set whose argument is clasfile is called outmm in the mortmorb macro.

classify(infile,outfile,dwod):

The macro classify augments the SAS^{TM} data set given in the argument infile with a variable named type. The type variable contains the sequence number of the censoring pattern defined in Tables 1 and 2 of Mihalko and Michalek (1998). The data set named in infile must contain the morbidity/mortality data as defined for the datafile argument of the mortmorb macro. The resulting SAS^{TM} data set will have the name given to the argument outfile.

In the macro mortmorb, the infile to classify is the clasfile named outmm from cellsgn. The outfile is also named outmm. The argument dwod is given the global macro variable dwod which was assigned a value in the macro initprob. At this point in the mortmorb macro, the original data file has been augmented twice. The current augmented version is named outmm and contains all the original data, the censoring type for each observation (in the variable named type), and the row and column for the observed and censoring values for each observation.

At this time, the only censoring types that can be correctly assigned by the classify macro are the 24 types listed in Mihalko and Michalek (1998). If the data includes other censoring types then this macro as well as the em macro that follows must be modified. The modification should be straightforward given the form of the code in these two macros.

em(datafile,probfile,emout,probout,numrows,numcols,total,tol, its,dwod):

The em macro performs the EM algorithm on observations in the SAS^{TM} data set named in the argument datafile. The datafile SAS^{TM} data set must contain all the variables described for the datafile argument in the mortmorb macro as well as row and column classifications for the observed and censoring values of the two time-to-event variables and a variable named type that contains the censoring type. The arguments for this macro are defined below.

probfile:

The argument probfile is a SAS^{TM} data set containing the initial probabilities for the time-to-onset of disease by time-to-death rectangles. The data set has one observation with total (the product of numrows and numcols) variables in a two dimensional array newp(&numrows,&numcols), where numrows and numcols are the number of rows and the number of columns for the matrix defining the rectangles, respectively.

emout:

The value given the argument emout is the name of a SAS^{TM} data set which will contain the original data in datafile and the EM algorithm distributions for each observation. The distributions for each observation are in the two dimensional array named ps(numrows,numcols). For each observation the double sum over all rows and all columns of the ps array should add to 1. The em macro performs this test after each iteration and puts a message into the log window if the distributions of the observations do not add up to the total sample size.

probout:

The argument probout holds the name to be given to the SAS^{TM} data set that contains the final set of probabilities for the time-to-onset of disease by time-to-death rectangles. The form of this data set is the same as that of the SAS^{TM} data set named in the argument probfile, including the same array prefix, newp. The values of this observation are the maximum likelihood estimates of the rectangle probabilities.

numrows, numcols, total, tol, its, dwod:

The arguments numrows, numcols and total are the number of rows, columns and rectangles in the matrix of rectangles. The arguments tol and its are the same as described in the main mortmorb macro. The argument dwod is the value 1 or 0 to indicate whether the data set contains observations where death is known to have occurred before disease.

How the em macro is used in the mortmorb macro:

The argument datafile is outmm, the twice augmented version of the original datafile. The data set of initial probabilities computed in the macro initprob and named initprob is used as the probfile argument. The argument emout is given the name that the user chose for the argument pseudo in the main macro, mortmorb. The user-specified name that was put into the argument outprobs in the main macro is used for the argument probout. The arguments numrows, numcols, total and dwod are those computed by initprob. The arguments tol and its are the values chosen by the user in the main macro.

revzero(probs,times,numrows,numcols,numtimes, probpref,newprobs,newtimes,dwod):

This macro checks for zero values for the possible rectangles. If there is a zero count in any possible rectangle then the matrix in probs is collapsed about that rectangle. After the collapsing, the search for zero rectangles is repeated. The process stops after the collapsing of the table eliminates all zero rectangles outside of the impossible rectangles. Each time the table is collapsed an endpoint defining the rectangles is eliminated. The new set of endpoints are put into a data set named in the argument newtimes. The newtimes SAS^{TM} data set has a one dimensional array with prefix t. A global macro variable named newtime is produced which contains the count of times in the data set newtimes. A global macro variable named yes is also produced. Yes is equal to 0 if the matrix of probabilities in the data set probs was not changed and equal to 1 if it was changed. The new set of probabilities is in a data set named by the value of the argument newprobs. The data set newprobs has one observation and a two dimensional array named p(newrows,newcols), where newrows and newcols are global macro variables which give the number of rows and columns in the new matrix of probabilities. If revzero changes the matrix of rectangles, a message will appear in the log window.

The arguments for revzero:

probs,probpref, numrows, numcols:

The argument probs holds one observation, which has a two dimensional array named probpref(numrows,numcols).

times, numtimes:

The argument times holds the name of a SAS^{TM} data set which contains the endpoints that define the intervals for the rectangles. Times must have one observation which has an array named t(numtimes).

newprobs, newtimes:

The arguments newprobs and newtimes are names chosen by the user for SAS^{TM} data sets to hold the new set of collapsed probabilities and the new set of interval endpoints that resulted from the collapsing of zero rectangles. Newprobs has one observation, which is a two dimensional array named probpref(newrows,newcols), where newrows and newcols are global macro variables produced by the macro. Newtimes has one observation with an array named t(newtimes), where newtimes is a macro variable giving the number of endpoints left after the collapsing.

dwod:

The argument dwod indicates with a 1 that there are observations for which death is known to have occurred before disease and it is 0 otherwise.

How the revzero macro is used in the mortmorb macro:

The revzero macro receives the matrix of probabilities produced by the macro em and collapses it around possible rectangles that have zero counts. This amounts to combining rectangles in the original time-to-onset by time-to-death matrix so that there are no possible rectangles that have an estimated probability of zero. Although zero is a legitimate estimated probability, a covariance matrix for the set of probabilities that contains a zero cannot be computed.

If the macro variable yes is 1, then it is neccessary to repeat the macros cellsgn and classify with the new rectangles and times. It is also neccessary to rewrite the formats defining the intervals for time-to-onset of disease and time-to-death. All this is done by the appropriate macros described earlier. The resulting rectangle memberships and censoring types are put into the macro em to compute maximum likelihood estimates for the probabilities of the new rectangles.

eminfo(pseudata,nrows,ncols,inform,rowcol):

The macro eminfo uses the method of Louis (1982) to estimate the information matrix for the estimates of the joint distribution of time-to-onset of disease and time-to-death using the final pseudodata produced by the EM algorithm. The arguments for this macro are:

pseudata, nrows, ncols:

The argument pseudata is the name of the data set that contains the final pseudodata produced by the EM algorithm. Each observation of the data set must contain two arrays, ps(nrows,ncols) and np(nrows,ncols), where the arguments nrows and ncols are the numbers of rows and columns of the time-to-onset by time-to-death array of rectangles. The array ps contains the contributions of each observation to each rectangle as determined by the EM algorithm. The array np contains the final EM algorithm estimates of the joint probabilities of the rectangles.

inform:

The value of the argument inform is the name of the output SAS^{TM} data set that contains the information matrix for the estimates of the joint probability distribution. This data set has one observation containing the estimate of the information matrix in a two dimensional array named im(infdim,infdim), where infdim is the number of rectangles with nonzero probabilities minus one. The dimension is one less than the number of possible rectangles since the probabilities must sum to 1. The observation also contains a one dimensional array named newp(infdim), which contains the vector of probabilities for the possible rectangles of the matrix. The data set inform also contains numrows, numcols and infdim, which are, respectively, the number of rows

and columns of the matrix of rectangles and the dimension of the information matrix.

The SAS^{TM} data set inform also contains the row and column numbers for possible rectangles in the matrix of rectangles. These numbers are contained in two one dimensional arrays ni(infdim) and nj(infdim), which are the row and column numbers respectively, that identify the vector newp of the inform data set with the original set of joint probabilities. For example, the probability newp(s) in the array newp of the output data set inform is assigned to the rectangle defined by the ni(s) row and the nj(s) column in the matrix of time-to-onset by time-to-death rectangles. This information is needed to identify estimates and their estimated variances computed from the newp and im arrays.

How the eminfo macro is used in the mortmorb macro:

The macro eminfo uses the emout data set named in the argument pseudo of the mortmorb declaration as the value of the argument pseudata. The values of the arguments nrows and ncols are the values of the global macro variables numrows and numcols as determined by the macro initprob (if there were no zero counts in the possible rectangles) or by the macro revzero (if some rectangles had to be combined). The information matrix is named by the mortmorb macro argument inform. The argument rowcol is renamed prefrowcol, where pref is the value of the argument pref in the argument list of the macro mortmorb.

The macro mm2:

The macro mm2 produces the same output that is produced by the macro mortmorb. The only difference between the two macro drivers is the method by which the initial rectangles are chosen. The macro mortmorb uses the macro eminvals and the user-defined argument intobs to determine the initial set of rectangles. In place of the macro eminvals, mm2 uses a macro named realfreq, which uses the formats, disfmt and dethfmt, to define the rectangles and produce the PROC FREQ output of counts of the uncensored data for these rectangles. This output is used by initprob to produce the initial probabilities for the EM algorithm in the mm2 macro. From this point forward, mm2 and mortmorb are the same.

How mortmorb and mm2 work together:

In order to compare the joint distributions for two samples, the distributions must be defined on the same set of rectangles. Mortmorb can be used on the smaller of the samples to obtain a set or rectangles and the joint probability and information matrix estimates. Mortmorb also produces the set of times and formats for the rows and columns of the matrix of rectangles. These data sets can be put into mm2 with the second sample to

obtain estimates for the second sample. Problems may arise if mm2 needs to combine some of the rectangles. Recall that if rectangles need to be combined by the revzero macro, a message will appear in the log window. In this case mm2 can be used again with the second sample rectangle formats and times to reclassify and estimate the joint distribution for the first sample. Another strategy is to predetermine the set of rectangles and run mm2 on both samples with formats and times defining the rectangles.

Summary of How the Macros Work:

Both macros, mortmorb and mm2, work basically the same way. The only difference is in the way in which the original rectangles are created. Mortmorb uses the uncensored data with a user-specified number of uncensored times between endpoints to create the rectangles, while mm2 uses rectangles determined by the user. The initial joint probabilities needed to begin the EM algorithm process are computed from the frequencies of the uncensored observations in the original rectangles. If there are no zero counts in the possible rectangles (i.e. those for which time-to-onset of disease is less than or equal to time-to-death or where death is known to have occurred before disease) then the initial probabilities are the sample proportions of the uncensored observations in the rectangles. If some of the possible rectangles have zero counts then those with zero counts are given a small positive count. The small amount added to a zero count rectangle is the larger of the sum of the counts for adjacent rectangles divided by the number of uncensored observations and the reciprocal of the number of uncensored observations. This assignment of initial probabilities is done by the macro initprob. The macro cellsgn first determines whether or not the data contain any observations for which death is known to have occurred before the disease. If this is the case, then the global macro variable dwod is assigned a value of one. Otherwise, dwod is assigned the value zero. Cellsgn then determines to which row and column of the matrix of rectangles the observed times or censoring times belong. These rectangle memberships are appended to the datafile. This appended datafile is the input for the macro classify, which determines each observation's censoring type. The censoring type for each observation is appended to the input file under the variable name type. The twice appended datafile is the input for the macro em, which uses the rectangle membership and censoring pattern information to distribute each observation among the rectangles. The EM algorithm stops when the sum of squared differences between successive joint probability estimates is less than the user-specified tolerance in the argument tol or when the number of interations exceeds the user-specified maximum number of iterations in the argument its. The array of distributions among the rectangles for each observation is the pseudodata. The macro revzero sums the contributions of each observation in the pseudodata array for each rectangle. If there are

any zero counts in the possible rectangles, then the macro collapses rows and columns around the zero rectangles until all of the possible rectangles have positive pseudocounts. The estimates of the joint probabilities are the sample proportions calculated from the pseudodata. If collapsing was necessary, these probabilities are used in the em macro as initial probabilities and a new set of pseudodata and estimates are calculated. The resulting probability estimates and the pseudodata are the input for the macro eminfo, which estimates the information matrix for the estimates of the joint probabilities using the method of Louis (1982).

The final probability estimates are output in the user-specified value of the argument outprobs in a one dimensional matrix newp. The information matrix is in a SAS^{TM} data set named by the user in the argument inform in a two dimensional matrix im. This data set also contains contains two one dimensional matrices named ni and nj which give the row and column numbers for the components of the probability estimate vector. A SAS^{TM} data set named by the user in the argument pseudo contains the resulting pseudodata as well as the final times that define the final rectangles, the number of rows and columns and the value of dwod in variables newrows, newcols and dwod. The formats for the row and column intervals that define the rectangles are in formats prefdis and prefdeth, where pref is a one character prefix chosen by the user in the main list of arguments, in the library named in the argument libdrive. Also in this library are two other formats named prefrow and prefcolumn, which format the row and column indices with the row and column interval definitions.

An Example Using Morbidity/Mortality Data

This example uses a predetermined format for the time-to-onset and time-to-death variables. The macro mm2 is run twice. First, it is run on the data for a group of exposed subjects with predefined formats. Then the four formats produced by this run of mm2 are put into mm2 using the control data. Examination of the log shows that the control run did not change the rectangles, so that all of the formats put out by mm2 in the two runs are the same. The data contains a variable Id that identifies an observation as that of a member of the exposed group (Id = E) or of the control group (Id = C). The variables used to create the censoring patterns, observations and censoring times are yotour1 (year of subject's entry into the environment, referred to as tour 1), yonset (year of onset), yod (year of death), mortstat (mortality status, mortstat = D means that the subject was already dead at the time this data was compiled) and cancer (cancer = 1 indicates the subject was known to have cancer, cancer = 0 means the subject was known not to have cancer, cancer = . means that the subject's cancer state was

unknown.).

The following SAS^{TM} data step computes time-to-onset of disease, t, and time-to-death, d, from year of entering the environment. The code also creates the new variables needed for the mortmorb and mm2 macros. This code is included under the name cancer.sas in the file named example. In order to run this example the file mortmorb must be put on the desktop and have the address $c:\windows\desktop\mortmorb$.

```
/***Cancer.sas****/
libname mm 'c:\windows\desktop\mortmorb';// data mm.cancer;// infile
'c:\windows\desktop\mortmorb\cancer.dat';// input
Casenr $1-5
Id $7 /* R=Exposed, C=Control*/
Race $9-10 /*NB=Non Black BL=Black*/
Occup 12 Yob 14-15
Mob 17-1
Yotour1 21-22
Motour1 24-25
Mortstat $27 /*D=Death*/
ica $28 /* Legend for Cause of Death*/
Yod 30-31
Mod 33-34
Cancer 36
Yonset 38-39
Pptr 41-47 /* Format 7.3*/
BodyFat 49-54 /* Format 6.3*/
;
run;
data mm.cancer;
set mm.cancer;
if yotour1 gt yonset and yonset ne . then delete;
if yonset gt yod and yod ne . then delete;
if yonset = . then t = .;
else t = yonset - yotour1; /*time to disease from tour1.*/
if yod = . then d = .;
else d = yod - yotour1; /*time to death from tour1.*/
current = 97 - yotour1; /*time from tour1 to present*/
if t = . and cancer = . and d = . then do; /*Type 23. No information
about cancer status and death is right censored.*/
obstl = .;
obstr = .;
```

```
obst = 0;
itl=0;
itr=1;
obsdl = .;
obsdr = .;
obsd = current;
idl = 0;
idr = 1;
end;
else if t = . and cancer = 0 and d = . then do; /*Type 19. Cancer is
known not to have occurred by the time death was right censored.*/
obstl = .;
obstr = .;
obst = .;
itl=0;
itr=1;
obsdl = .;
obsdr = .;
obsd = current;
idl = 0;
idr = 1;
end; else if t ne . and d = . then do; /*Type 5. Time-to-onset is known
and time-to-death right censored.*/
obstl = .;
obstr = .;
obst = t;
itl=0;
itr=0;
obsdl = .;
obsdr = .;
obsd = current;
idl = 0;
idr = 1;
end;
else if t = . and cancer = . and d ne . then do; /*Type 21. No information
about disease and time-to-death is known. */
obstl = .; obstr = .;
obst = 0;
itl=0;
itr=1;
obsdl = .;
obsdr = .;
```

```
obsd = d;
idl = 0;
idr = 0;
end;
else if t = . and cancer = 0 and d ne . then do; /*Type 17. Cancer is
known not to have occurred before death and time-to-death known.*/
obstl = .;
obstr = .;
obst = .;
itl=0;
itr=1;
obsdl = .;
obsdr = .;
obsd = d;
idl = 0;
idr = 0;
end;
else if t = . and cancer = 1 and d = . then do; /*Type 8. Cancer is known
to have occurred before and time-to-death is right censored.*/
obstl = .;
obstr = .;
obst = current;
itl=1;
itr=0;
obsdl = .;
obsdr = .;
obsd = current;
idl = 0;
idr = 1;
end;
else if t ne . and d ne . then do; /*Type 1. Both time-to-onset and
time-to-death are known.*/
obstl = .;
obstr = .;
obst = t;
itl=0;
itr=0;
obsdl = .;
obsdr = .;
obsd = d;
idl = 0;
idr = 0;
```

```
end;
run;
```

The next data step divides the data set into exposed and controls. This file is named split.sas in the example folder.

```
Data mm.exposed mm.control
set mm.cancer;
if id = 'E' then output mm.exposed;
else output mm.control;
run;
```

The next step is to create the original formats. In this example we use five year intervals. Because there were very few events in the first ten years, the first interval is zero to fifteen. The SAS^{TM} code below sets up the two formats for time-to-onset and time-to-death as well as the endpoints needed as input to the mm2 macro. This code is in Yr5fmt.sas in the example folder.

```
libname library 'c : \windows\desktop\mortmorb';
proc format library = library;
value dis . = 'death wo disease'
0-15 = '[0,15)'
15-20 = '[15,20)'
20-25 = '[20,25)'
25-30 = '[25,30)'
30-35 = '[30,35)'
35-high = '[35,+)';
run;

proc format library = library;
value deth 0- 15 = '[0,15)'
15- 20 = '[15,20)'
20- 25 = '[20,25)'
25 -30 = '[25,30)'
30- 35 = '[30,35)'
35- high = '[35,+)';
run;

data times;
```

```
input t1-t5;
cards;
15 20 25 30 35
;
run;
```

We are now ready to run mm2 on the data. The following code, called Fixboth.sas in the example folder, runs mm2 on the exposed data using the the formats and times just created. The macro mm2 then puts out the set of times with four formats for the time-to-disease and time-to-death intervals. These possibly new times and formats rdis and rdeth are put into mm2 with the control data. The log does not indicate that the times needed to be changed on this second run, so that we know the four formats cdis, cdeth, crow and ccolumn are the same as the corresponding ones that apply to the exposed cohort.

```
/***Fixboth.sas*****/

options ls=78 nonotes;
libname mm 'c:\windows\desktop\mortmorb';
%include 'c:\windows\desktop\mm2.sas';
%mm2(mm.exposed,mm.rprobs,mm.rpseu,dis,deth,times,.0001,100,
c:\windows\desktop\mortmorb,mm.rinform,r);
%mm2(mm.control,mm.cprobs,mm.cpseu,rdis,rdeth,times, .0001,100,
c:\windows\desktop\mortmorb,mm.cinform ,c);
```

We now have estimates of the joint probabilities for a set of time-to-onset of disease and time-to-death variables and their information matrices for both groups. The data sets mm.rinform and mm.cinform contain both the information matrices and the vector of the first K−1 out of K (in our case, 13 out of 14) final joint probability estimates. In addition, these inform data sets contain arrays named Ni and Nj of length K−1 each, holding the original coordinates of the probabilities. This information is all that is needed to compare the joint probability distributions. In order to compute and compare functions of the joint probabilities we need to complete the vector of joint probabilities and compute the covariance matrices for the full set of probabilities. We will then be ready to use PROC IML of SAS^{TM} to compute functions of the joint probabilities and the covariance structures of these functions.

The following SAS^{TM} code receives the output of the two mm2 runs and prepares it for use with PROC IML. This code is in the file named

Prepjont.sas in the example folder. This code also prints the tables of joint probabilities and their standard deviations for the exposed and the control groups. It also prints a side-by-side table of the joint probabilities with a title that gives the results of the chi-square test of equality for the two distributions. The data sets rfilld and cfilld contain the vector of probability estimates and the complete covariance matrix and the coordinates for the original rectangle for each estimate.

```
/***prepjont.sas**/

%include 'c:\windows\desktop\mortmorb\matrx.sas';
%include 'c:\windows\desktop\mortmorb\estcov.sas';
%include 'c:\windows\desktop\mortmorb\fillout.sas';
%include 'c:\windows\desktop\mortmorb\quaddiff.sas';

%matrx(mm.rinform,rmats);

%matrx(mm.cinform,cmats); /**Puts information matrices into matrix inform.*/
%estcov(rmats,rcov);
%estcov(cmats,ccov);/*creates covariance matrix from information.*/
%pstdtab(rcov,rrow,rcolumn,10,8.6,Exposed Joint Probabilities);
%pstdtab(ccov,crow,ccolumn,10,8.6,Control Joint Probabilities);
/*Joint probabilities and their standard deviations
are tabled in these last two calls.**/
%fillcov(rcov,rfilld);
%fillcov(ccov,cfilld); /*puts out the complete covariance matrix
for all probabilities.*/
%quaddiff(rfilld,cfilld,rcolumn,jointprobs,Exposed,Controls);
/*Compares the joint probability distributions*/
```

We now have two data sets, rfilld and cfilld, containing the joint probability distributions and their covariance martrices for the exposed and control groups on the same set of rectangles. These data sets are in a form that allows PROC IML to be used on them directly. We can now use these data sets to compute a maximum likelihood estimate of any function of the the joint distributions as well as the covariance structure of the function. As two examples of doing this we compute relative risk and lethality.

The following code computes the relative risk of death with cancer for the exposed versus the controls. The results are displayed and discussed in

Mihalko and Michalek (1998). The purpose here is to show how some of the macros in the mortmorb folder are used. We compute the relative risk of the a subject's morbidity/mortality experience being in a particular rectangle, the relative risk of dying with disease in a particular time-to-death interval, and the overall relative risk of dying with disease.

The code below is in the file named allRR.sas in the example file. This file is a driver which calls up three other macros listed in the include statement. Each of these macros calls other macros that do the following. First, the macro determines which joint probabilities are needed. It then calls a macro which extracts these probabilities and the corresponding covariance matrix from the input file rfilld or cfilld. Other macros compute sums and ratios of the probabilities. Both relative risk and lethality are ratios of sums of probabilities from the joint distributions. For each function, a special macro is needed to apply the delta method for finding the covariance structure of the function being computed. These macros also call up the tabling macro, which tables the estimates and their standard deviations in terms of the original rectangles or columns, whichever is appropriate.

```
/*******allRR.sas*******/
libname library 'c:\windows\desktop\mortmorb';
%include 'c:\windows\desktop\mortmorb\relrskij.sas';
%include 'c:\windows\desktop\mortmorb\colrsk.sas';
%include 'c:\windows\desktop\mortmorb\totrlrsk.sas';

options nonotes;
%relrisk(rfilld,cfilld,Relative Exposed to Control);
%let t = Relative Risk of dying with disease Exposed to Control;
%colrsk(rfilld,cfilld,rcolumn,final,&t);
%totrlrsk(rfilld,cfilld,final, Total Relative Risk of Dying with disease);
```

Lethality by column is tabled for both groups. The heading of the table gives the results of a chi-square test for the equality of the vectors of lethalities between the two groups.

References:

Louis TA (1982). Finding the observed information matrix when using the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 44, 226-233.

Mihalko D and Michalek JE (1998). *A Non-Parametric Analysis of Morbidity/Mortality Data*, Final Technical report for IPA-MIHALKO 96-98.

Appendix 2: SAS^{TM}Macros for Non-parametric Estimation of the Joint Probability Distribution of Time-to-Onset of Disease and Time-to-Death

```
/*******allRR.sas*******/
/****This code uses output from prepjont to get all three versions****
*****of relative risk.    **/
libname library 'c:\windows\desktop\mortmorb';/* where formats are stored*/
%include 'c:\windows\desktop\mortmorb\relrskij.sas';
%include 'c:\windows\desktop\mortmorb\colrsk.sas';
%include 'c:\windows\desktop\mortmorb\totrlrsk.sas';
options nonotes;
%relrisk(rfilld,cfilld,Relative Risk Exposed to Controls);
%let t = Relative Risk of dying with disease Exposed to Controls;
%colrsk(rfilld,cfilld,rcolumn,final,&t);
%totrlrsk(rfilld,cfilld,final, Total Relative Risk of Dying with disease);
/***Cancer.sas****/
libname mm 'c:\windows\desktop\mortmorb';
data mm.cancer;
infile 'c:\windows\desktop\mortmorb\cancer.dat';
input
Casenr      $1-5
Id          $7          /* E=Exposed, C=Controls */
Race        $9-10       /*NB=Non Black BL=Black*/
Occup        12         /*1,2=Flying Officers 3=Non-Flying Officers
                        4=Flying Enlisted 5=Non-Flying Enlisted */
Yob          14-15
Mob          17-19
Yotour1      21-22
Motour1      24-25
Mortstat    $27         /*D=Death*/
ica         $28         /* Legend for Cause of Death*/
Yod          30-31
Mod          33-34
Cancer       36
Yonset       38-39
Pptr         41-47          /* Format 7.3*/
BodyFat      49-54          /* Format 6.3*/
;
run;

data mm.cancer;
set mm.cancer;
```

```
if yotour1 > yonset and yonset ne . then delete;
if yonset > yod and yod ne . then delete;
if yonset = . then t = .;
else t = yonset - yotour1;  /*time to disease from tour1.*/
if yod = . then d = .;
else d = yod    - yotour1;  /*time to death from tour1.*/
current = 97 - yotour1; /*time from tour1 to present*/
if t = . and cancer = . and d = . then do;  /*Type 23.  No information
                                              ***about cancer status and
                                              ***death is right censored.*/
          obstl = .;
          obstr = .;
          obst  = 0;
          itl=0;
          itr=1;
          obsdl = .;
          obsdr = .;
          obsd  = current;
          idl = 0;
          idr = 1;
end;
else if t = . and cancer = 0 and d = . then do;  /*Type 19.  Cancer is
                                                   ***known not to have
                                                   ***occurred by the time
                                                   ***death was right
                                                   ***censored.*/
          obstl = .;
          obstr = .;
          obst  = .;
          itl=0;
          itr=1;
          obsdl = .;
          obsdr = .;
          obsd  = current;
          idl = 0;
          idr = 1;
end;
else if t ne . and d = . then do; /*Type 5.  Time-to-onset is known
                                    ***and death time right censored.*/
          obstl = .;
          obstr = .;
          obst  = t;
```

```
            itl=0;
            itr=0;
            obsdl = .;
            obsdr = .;
            obsd  = current;
            idl = 0;
            idr = 1;
end;
else if t = . and cancer = . and d ne . then do;  /*Type 21.  No information
                                                    ***about disease and
                                                    death time is known. */
            obstl = .;
            obstr = .;
            obst  = 0;
            itl=0;
            itr=1;
            obsdl = .;
            obsdr = .;
            obsd  = d;
            idl = 0;
            idr = 0;
end;
else if t = . and cancer = 0 and d ne . then do;  /*Type 17. Cancer is
                                                   ***known not to have
                                                   ***occurred before death
                                                   and death time known.*/
            obstl = .;
            obstr = .;
            obst  = .;
            itl=0;
            itr=1;
            obsdl = .;
            obsdr = .;
            obsd  = d;
            idl = 0;
            idr = 0;

end;
else if t = . and cancer = 1 and d = . then do;  /*Type 8.  Cancer is known
                                                  ***to have occurred before
                                                  ***and death time is right
                                                  ***censored.*/
```

```
            obstl = .;
            obstr = .;
            obst  = current;
            itl=1;
            itr=0;
            obsdl = .;
            obsdr = .;
            obsd  = current;
            idl = 0;
            idr = 1;
end;
else if t ne .   and d ne . then do;  /*Type 1.  Both time-to-onset and
                                      ***time-to-death are known.*/
            obstl = .;
            obstr = .;
            obst  = t;
            itl=0;
            itr=0;
            obsdl = .;
            obsdr = .;
            obsd  = d;
            idl = 0;
            idr = 0;
end;
run;
/*****cellsgn2.sas  3/30/98**/
%macro cellsgn(infile,datafile,clasfile,numrow,numcol,dwod);
/*********************************************************************
***Infile contains the times that make up the disease and death   ***
***time lattice.                                                  ***
***datafile is the file containing the mortality morbidity data.  ***
***clasfile will be the datafile augmented with obst, obstl and   ***
***obstr classified by row and obsd, obsdl and sbsdr classified by***
***column.  numrow and numcol are the numbers of row and columns, ***
***respectively. dwod is 1 if there is a first row of missing     ***
***disease times and 0 otherwise.                                 ***
*********************************************************************/
%let numt = %eval(&numrow -1-&dwod);
%let numd = %eval(&numcol - 1);
data null;
set &infile;
%do i = 1 %to &numt;
```

```
call symput("t&i",trim(left(t&i)));
%end;
stop;
run;
data null;
set &infile;
%do i = 1 %to &numd;
call symput("d&i",trim(left(t&i)));
%end;
stop;
run;
/**The next data step classifies the disease and death times into***
***respective column and row intervals.  Originally, pairs with  ***
***death preceding disease are assigned to row = 0.  At the end  ***
***of the data step, if there are such observations, i.e. dwod=1 ***
***then dwod (which will be 1 in that case) is added to each row ***
***number.                                                       **/
data &clasfile;
set &datafile;
    if obsd <= &d1 then column =  1;
     if obsdl <= &d1 then columnl =  1;
     if obsdr <= &d1 then columnr =  1;
%do i = 2 %to &numd ;
          %let ii = %eval(&i - 1);
         if  &&d&ii< obsd <= &&d&i then column = &i;
         if  &&d&ii< obsdl <= &&d&i then columnl = &i;
         if  &&d&ii< obsdr <= &&d&i then columnr = &i;
%end;
         if obsd > &&d&numd then column = &numcol;
         if obsdl > &&d&numd then columnl = &numcol;
         if obsdr > &&d&numd then columnr = &numcol;
  if obstr = 0 then do;
           row = 1;       /**this is for totally unknown disease info.*/
           rowr = 0;      /*which are types 21 through 24. */
           rowl = 0;
  end;
  else do;
     if obst = . then row = 0;
     else if obst <= &t1 then row = 1;
     if obstl = . then rowl = 0;
     else if obstl <= &t1 then rowl = 1;
     if obstr = . then rowr = 0;
```

```
     else if obstr <= &t1 then rowr = 1;
  end;
%do i = 2 %to &numt;
          %let ii = %eval(&i - 1);
         if    &&t&ii < obst <= &&t&i then row = &i;
         if    &&t&ii < obstl <= &&t&i then rowl = &i;
         if    &&t&ii < obstr <= &&t&i then rowr = &i;
%end;
         if     obst > &&t&numt then row = %eval(&numt+1);
         if     obstl > &&t&numt then rowl = %eval(&numt+1);
         if     obstr > &&t&numt then rowr = %eval(&numt+1);
row = row + &dwod;
rowl = rowl + &dwod;
rowr = rowr +&dwod;
run;
%mend;
/*cellsgn(infile,datafile,clasfile,numrow,numcol)*/
/**numrows and numcols come from intprob2**/
/*%cellsgn(times,mortmorb,outmm,&numrows,&numcols);*/
/***classify.sas 4/21/98**/

/*****************************************************
***In order to implement the em algorithm using   ***
***the initial probabilities produced by test.sas ***
***we need to classify each observation in the    ***
***original data set by censoring type.  This     ***
***classification will involve a variable which   ***
***gives the type number and the interval values  ***
***for the censored disease and death times.      ***
***We will also append the array of variables     ***
***to each observation to hold the pseudo data    ***
***contributions of each observation to each      ***
***interval.                                      ***
***Also appended to each observation will be the  ***
***current set of joint probabilities for disease ***
***and death.  The censoring type will and the    ***
***censored values will determine which           ***
***probabilities get put into the pseudo-data     ***
***cells.                                         ***
*****************************************************/

%macro  classify(infile,outfile,dwod);
```

```
/*****************************************************
***Infile contains mortality/morbidity data of the***
***form built by whodis. Outfile will contain the ***
***same data with another variable called type,   ***
***which will indicate censoring type.            ***
***The following table defines the censoring types***
***that this macro will handle.                   ***
***dwod indicates by a 1 if there are cases of    ***
***known deaths without disease and is 0 otherwise***
*****************************************************/

/********Attempt to classify mortality morbidity data by ********
*********data type. */
%if %eval(&dwod) = 1 %then %do;
data dandd justdeth;
set &infile;
if (obst=. and obstl=.) /*Death info known to be before disease.*/
      or (itL= 0 and itR = 1 and obst = 0 ) /*no disease info*/
      then output justdeth;
else output dandd;
run;
%end;
%else %do;
data dandd;
set &infile;
run;
%end;
/***dandd are the observations who were known to have the
****disease before death.  Justdeth speaks for itself.*/
/*Now we use the indicators to classify each data type*/
data dandd;
set dandd;
if itl=0 and itr=0 and idl=0 and idr = 0 then type = 1;
else if itl=1 and itr=0 and idl=0 and idr = 0 then type = 2;
else if itl=0 and itr=1 and idl=0 and idr = 0 then type = 3;
else if itl=0 and itr=0 and idl=1 and idr = 0 then type = 4;
else if itl=0 and itr=0 and idl=0 and idr = 1 then type = 5;
else if itl=1 and itr=1 and idl=0 and idr = 0 then type = 6;
else if itl=1 and itr=0 and idl=1 and idr = 0 then type = 7;
else if itl=1 and itr=0 and idl=0 and idr = 1 then type = 8;
else if itl=0 and itr=1 and idl=1 and idr = 0 then type = 9;
```

```
else if itl=0 and itr=1 and idl=0 and idr = 1 then type = 10;
else if itl=0 and itr=0 and idl=1 and idr = 1 then type = 11;
else if itl=1 and itr=1 and idl=1 and idr = 0 then type = 12;
else if itl=1 and itr=1 and idl=0 and idr = 1 then type = 13;
else if itl=1 and itr=0 and idl=1 and idr = 1 then type = 14;
else if itl=0 and itr=1 and idl=1 and idr = 1 then type = 15;
else if itl=1 and itr=1 and idl=1 and idr = 1 then type = 16;
run;
%if %eval(&dwod) = 1 %then %do;
data justdeth;
set justdeth;
if obst = . then do;
     if idl=0 and idr=0 then type=17;
     else if idl=1 and idr=0 then type=18;
     else if idl=0 and idr=1 then type=19;
     else if idl=1 and idr=1 then type=20;
end;
else if obst = 0 then do;
     if idl=0 and idr=0 then type=21;
     else if idl=1 and idr=0 then type=22;
     else if idl=0 and idr=1 then type=23;
     else if idl=1 and idr=1 then type=24;
end;
run;
data &outfile;
set dandd justdeth;
run;
%end;
%else %do;
data &outfile;
set dandd;
run;
%end;
%mend;
/*classify(infile,outfile,dwod);*/
/*%classify(outmm,outmm,1);*/
/***classify.sas 4/21/98**/
/*****************************************************
***In order to implement the em algorithm using   ***
***the initial probabilities produced by test.sas ***
***we need to classify each observation in the    ***
***original data set by censoring type.  This     ***
```

```
***classification will involve a variable which   ***
***gives the type number and the interval values  ***
***for the censored disease and death times.      ***
***We will also append the array of variables     ***
***to each observation to hold the pseudo data    ***
***contributions of each observation to each      ***
***interval.                                      ***
***Also appended to each observation will be the  ***
***current set of joint probabilities for disease ***
***and death.  The censoring type will and the    ***
***censored values will determine which           ***
***probabilities get put into the pseudo-data     ***
***cells.                                         ***
*************************************************/

%macro  classify(infile,outfile,dwod);

/*************************************************
***Infile contains mortality/morbidity data of the***
***form built by whodis. Outfile will contain the ***
***same data with another variable called type,   ***
***which will indicate censoring type.            ***
***The following table defines the censoring types***
***that this macro will handle.                   ***
***dwod indicates by a 1 if there are cases of    ***
***known deaths without disease and is 0 otherwise***
*************************************************/

/********Attempt to classify mortality morbidity data by ********
*********data type. */
%if %eval(&dwod) = 1 %then %do;
data dandd justdeth;
set &infile;
if (obst=. and obstl=.) /*Death info known to be before disease.*/
      or (itL= 0 and itR = 1 and obst = 0 ) /*no disease info*/
            then output justdeth;
else output dandd;
run;
%end;
%else %do;
data dandd;
set &infile;
```

```
run;
%end;
/***dandd are the observations who were known to have the
****disease before death.  Justdeth speaks for itself.*/
/*Now we use the indicators to classify each data type*/
data dandd;
set dandd;
if itl=0 and itr=0 and idl=0 and idr = 0 then type = 1;
else if itl=1 and itr=0 and idl=0 and idr = 0 then type = 2;
else if itl=0 and itr=1 and idl=0 and idr = 0 then type = 3;
else if itl=0 and itr=0 and idl=1 and idr = 0 then type = 4;
else if itl=0 and itr=0 and idl=0 and idr = 1 then type = 5;
else if itl=1 and itr=1 and idl=0 and idr = 0 then type = 6;
else if itl=1 and itr=0 and idl=1 and idr = 0 then type = 7;
else if itl=1 and itr=0 and idl=0 and idr = 1 then type = 8;
else if itl=0 and itr=1 and idl=1 and idr = 0 then type = 9;
else if itl=0 and itr=1 and idl=0 and idr = 1 then type = 10;
else if itl=0 and itr=0 and idl=1 and idr = 1 then type = 11;
else if itl=1 and itr=1 and idl=1 and idr = 0 then type = 12;
else if itl=1 and itr=1 and idl=0 and idr = 1 then type = 13;
else if itl=1 and itr=0 and idl=1 and idr = 1 then type = 14;
else if itl=0 and itr=1 and idl=1 and idr = 1 then type = 15;
else if itl=1 and itr=1 and idl=1 and idr = 1 then type = 16;
run;
%if %eval(&dwod) = 1 %then %do;
data justdeth;
set justdeth;
if obst = . then do;
     if idl=0 and idr=0 then type=17;
     else if idl=1 and idr=0 then type=18;
     else if idl=0 and idr=1 then type=19;
     else if idl=1 and idr=1 then type=20;
end;
else if obst = 0 then do;
     if idl=0 and idr=0 then type=21;
     else if idl=1 and idr=0 then type=22;
     else if idl=0 and idr=1 then type=23;
     else if idl=1 and idr=1 then type=24;
end;
run;
data &outfile;
set dandd justdeth;
```

```
run;
%end;
%else %do;
data &outfile;
set dandd;
run;
%end;
%mend;
/*classify(infile,outfile,dwod);*/
/*%classify(outmm,outmm,1);*/
%macro colrsk(numcovp,dencovp,colfmt,outfile,title);
/*******************************************************
***This macro takes two matrices of estimated joint  ***
***probabilities and produced a table of relative    ***
***risks with their standard deviations.             ***
***numcovp is a SAS data set for the numerator of the***
***relative risk.  dencovp is a SAS data set for the ***
***denominator of the relative risks.  Both SAS data ***
***sets must have the same number or rows and columns***
***corresponding to the same set of rectangles in a  ***
***two dimensional matrix.  The columns for the data ***
***sets must be variables named cov1-covK,p, row and ***
***column.  K is the number of joint non-zero joint  ***
***probabilities for the rectangles and also the     ***
***number of rows.  Cov1-covK are the columns of the ***
***covariance matrix for the estimates of the K joint***
***probabilities contained in variable p.  Row and   ***
***column are the row and column indices, indicating ***
***the coordinates of the rectangle for p is the     ***
***estimate of the joint probability.  The row and   ***
***column values for both data sets must be the same ***
***and in the same order.  Colfmt is the format that ***
***gives the column intervals defined by the column  ***
***values.                                           ***
***A SAS data set given the name in the argument     ***
***outfile will contain number of observations equal ***
***to the number of columns with variables relrisk,  ***
***stdev and column.                                 ***
***Title will be the title of the printed output.    ***
*******************************************************/

/**WARNING!!!   WARNING!!!    WARNING!!!   WARNING!!!***
```

```
*** Be sure that the follwoing include statements    ***
***have the correct addresses.                        ***
***WARNING!!!   WARNING!!!    WARNING!!!   WARNING!!!**/
      %include 'c:\windows\desktop\mortmorb\submtrx.sas';
      %include 'c:\windows\desktop\mortmorb\sumstd.sas';
      %include 'c:\windows\desktop\mortmorb\ratiostd.sas';
/***The following code gets the dimension of the pvector for the two***
****groups and tests to see if they are equal.                      ***
*************************************************************************/
data _null_;
set &numcovp nobs=n end = last;
call symput('rdim',compress(n));
if last then do;
   call symput('colnum',compress(column)); /**number of columns put into
                                            macro variable colnum**/
   call symput('rownum',compress(row)); /**number rows in rownum**/
end;
run;
data _null_;
set &dencovp nobs=n;
if n = &rdim then do;
     end = 0;
     call symput('end',compress(end));
     stop;
end;
else do;
     end = 1;
     call symput('end',compress(end));
     stop;
end;
run;
%if &end = 1 %then %do;
  %put  The dimensions of the two probability vectors are unequal.;
  %put  We cannot compute the requested table.;
%end;
%else %do;
data covnum;   /****r is the prefix for the numerator.**/
set &numcovp;
array cov{&rdim};
array rcov{&rdim};
%do i = 1 %to &rdim;
      rcov(&i) = cov(&i);
```

```
%end;

      rrow = row;
      rcolumn = column;
run;
data covden;        /***c is the prefix for the denominator.**/
set &dencovp;
array cov{&rdim};
array ccov{&rdim};
%do i = 1 %to &rdim;
      ccov(&i) = cov(&i);
%end;
      crow = row;
      ccolumn = column;
run;
data bothcovp;
merge covnum covden;
run;
/***testing to be sure the row and column indices match.***/
data _null_;
set bothcovp ;
if rrow ne crow or rcolumn ne ccolumn then do;
      end = 1;
      call symput('end',compress(end));
      stop;
end;
run;
%if &end = 1 %then %do;
   %put The row and/or column indices for the two vectors;
   %put Do not match.;
   %put We must stop here.;
%end;
%else %do;
      data covnum;
      set covnum;
      row=rrow;
      column = rcolumn;
      drop rrow rcolumn rcov1-rcov&rdim ;
      run;
      data covden;
      set covden;
      row=crow;
```

```
      column = ccolumn;
      drop crow ccolumn ccov1-ccov&rdim;
      run;
   %do j = 1 %to &colnum;
       /****Now we pull out the probabilities and covariance****
       *****submatrix needed to compute the numerator and   ****
       *****denominator for the relative risk of dying with ****
       *****disease in column j.  It is assumed that row 1  ****
       *****death without disease. So that the probability  ****
       *****of dying with disease in column j is the sum of ****
       *****the joint probabilties in column j from row 2 to****
       *****the min of rownum and (j+1).                    ****
       ********************************************************/
      data _null_;
      maxrow = &j + 1;
      lastrow = min(maxrow,&rownum);
      call symput('lastrow',compress(lastrow));
      stop;
      run;
      %submtrx(covnum,2,&j,&lastrow,&j,numjcov);
      %submtrx(covden,2,&j,&lastrow,&j,denjcov);
      %sumstd(numjcov,cov,numout);
      %sumstd(denjcov,cov,denout);

/**The files numout and denout contain one observation each***
***with variables sum and stdev, which are the sum of the  ***
***probabilities in the column and the standard deviation  ***
***of the sum, respectively.  The next code puts these     ***
***quantities into macro variables with the appropriate    ***
***names.                                                  ***
************************************************************/

      data _null_;
      set numout;
      sig = stdev*stdev;
      call symput('numsig',compress(sig));
      call symput('numsum',compress(sum));
      stop;
      run;

      data _null_;
      set denout;
```

```
      sig = stdev*stdev;
      call symput('densig',compress(sig));
      call symput('densum',compress(sum));
      stop;
      run;

/***The next call to ratiostd computes the relative risk for row j***
****along with its standard deviation.                              ***
**********************************************************************/

      %ratiostd(&numsum,&densum,&numsig,&densig,0,ratout);

/***Now the relative risk and standard deviation will be put into***
****the output data set.                                            ***
**********************************************************************/
   %if &j = 1 %then %do;
      data &outfile;
      label relrisk = 'Relative Risk' stdev = 'Standard Deviation';
      set ratout;
      relrisk = ratio;
      column = 1;
      keep column relrisk stdev;
      run;
   %end;

   %else %do;
      data ratout;
      set ratout;
      relrisk = ratio;
      column = &j;
      keep column relrisk stdev;
      run;

      data &outfile;
      label relrisk = 'Relative Risk' stdev = 'Standard Deviation'
            column = 'Time-to-Death';
      set &outfile ratout;
      run;
   %end;

    %end;
      proc tabulate data = &outfile format=f10.6;
```

```
      format column &colfmt..;
      class column;
      var relrisk stdev;
      table column, sum*(relrisk stdev);
      keylabel sum = '  ';

      title "&title";
      run;

  %end;
%end;
%mend;

/*example*/
/*colrsk(numcovp,dencovp,colfmt,outfile,title)*/
/*******example*******/
/*libname library 'c:\windows\desktop\mortmorb'; where formats are stored*/
/*options nonotes;
%let t = Relative Risk of dying with disease Exposed to Controls;
%colrsk(rfilld,cfilld,rcolumn,final,&t); */
/**em.sas  3/30/98**/
/*Uses the output of cellsgn, newtimes5 and the array numbering
***system of sas along with the data type combined with &probfile
***to determine how to allocate each observation to the pseudo
***data array.**/
/********This data step puts the cell probabilities into
***each observation of the mortmorb data set with the
***row and column classification of &datafile.  *****/
%macro em(datafile,probfile,emout,probout,numrows,numcols,total,tol,its
                                                              ,dwod);

/*************************************************************************
***This macro performs the em algorithm on the mortality morbidity   ***
***data contained in datafile.  Data file must have a variable named ***
***type that indicated what is the censoring type for each           ***
***observation.  Censoring types are assigned according to the macro ***
***classify which uses the 24 censoring types from Mihalko & Michalek***
***(1998).  Each observation must also be classified according to    ***
***the row and column of the cell of the disease by death matrix     ***
***defined by the matrix of probabilities in probfile.  Normally     ***
***datafile will be the output file from the macro classify.         ***
***probfile must contain &total probabilities than sum to 1          ***
```

```
***and have names newp1 to newp&total as SAS numbers numrows by      ***
***numcols array.  tol is the convergence check.                     **/
/*********************************************************************
***The pseudo data are computed by summing the probabilites of each  ***
***cell that the censoring pattern allows as a possibility and using ***
***the sum as the denominator for the conditional cell probabilities ***
***computed by dividing the denominator into the cell probability    ***
***for each possible cell.  The initial probabilities are those in   ***
***probfile.  Probfile is updated after all the observations have    ***
***been distributed according to their possible cell memberships by  ***
***computing new cell probabilities using the sample proportions of  ***
***the current pseudodata.                                           ***
*********************************************************************/
data nextprob;
set &probfile;
run;
%let ind = 0;
%let i = 1;
%do %while(&ind < 1 AND &i <= &its );
data &emout;
merge &datafile nextprob;
array newp{&numrows,&numcols};
array np{&numrows,&numcols};
if _n_ = 1 then do;
        do i = 1 to &numrows;
            do j = 1 to &numcols;
                np(i,j) = newp(i,j);
            end;
        end;
end;
retain np1 - np&total;
drop newp1 - newp&total i j;
run;

/**In the following code the column-1 and row+1 are due to the
***fact that row 1 is disease missing row.  So that row i is the
***same as column(i-1).**/

data &emout;
set &emout;
array np{&numrows,&numcols};
array ps{&numrows,&numcols}; /**These will contain the pseudodata
```

```
                                    contributions.*/
  denom = 0;
 do i = 1 to &numrows;
                do j = 1 to &numcols;
                  ps(i,j) = 0;
                end;
              end;
if type = 1 then do;

       ps(row,column) = 1;
 end;
else if type = 2 then do;

          do i = 1 + &dwod to row;
               denom = denom + np(i,column);
            end;
            do i = 1 +  &dwod to row;
             if denom = 0 then ps(i,column) = 0;
              else  ps(i,column) = np(i,column)/denom;
            end;
end;
else if type = 3 then do;

          do i = row to min(&numrows,column+ &dwod);
               denom = denom + np(i,column);
            end;
          do i = row to min(&numrows,column+ &dwod);
            if denom = 0 then ps(i,column) = 0;
             else
               ps(i,column) = np(i,column)/denom;
          end;
end;

else if type = 4 then do;
          do j = row- &dwod to min(&numrows- &dwod,column);
               denom = denom + np(row,j);
            end;
          do j = row- &dwod to min(&numrows- &dwod,column);
              if denom = 0 then ps(row,j) = 0;
               else ps(row,j) = np(row,j)/denom;
            end;
end;
```

```
else if type = 5 then do;
          do j = max(row- &dwod,column) to &numcols;
             denom = denom + np(row,j);
          end;
          do j = max(row- &dwod,column) to &numcols;
            if denom = 0 then ps(row,j)=0;
           else  ps(row,j) = np(row,j)/denom;
          end;
end;

else if type = 6 then do;
          do i = rowl to min(rowr,column+ &dwod);
             denom = denom + np(i,column);
          end;
          do i = rowl to min(rowr,column+ &dwod);
            if denom = 0 then ps(i,column) =0;
            else ps(i,column) = np(i,column)/denom;
          end;
end;

else if type = 7  then do;
         do i = 1+ &dwod to row;
           do j = i- &dwod to column;
               denom = denom + np(i,j);
           end;
         end;
         do i =1 + &dwod to row;
           do j = i- &dwod to column;
             if denom = 0 then ps(i,j) = 0;
             else  ps(i,j) = np(i,j)/denom;
           end;
         end;
end;

else if type = 8 then do;
         do i = 1 + &dwod to min(row,column+ &dwod);
           do j = max(row- &dwod,column) to &numcols;
              denom = denom + np(i,j);
           end;
         end;
         do i = 1+ &dwod to min(row,column+ &dwod);
```

```
            do j = max(row- &dwod,column) to &numcols;
             if denom = 0 then ps(i,j) = 0;
            else ps(i,j) = np(i,j)/denom;
            end;
          end;
end;

else if type = 9 then do;
          do i = row to min(&numrows,column+ &dwod);
                do j = i- &dwod to column;

                  denom = denom + np(i,j);

                end;
          end;

          do i = row to min(&numrows,column+ &dwod);
                do j = i- &dwod to column;
                 if denom = 0 then ps(i,j) = 0;
                 else ps(i,j) = np(i,j)/denom;
                end;
          end;
end;

else if type = 10 then do;
          do i = row to &numrows;
            do  j= max(i- &dwod,column) to &numcols;
              denom = denom + np(i,j);
            end;
          end;
           do i = row to &numrows;
            do  j= max(i- &dwod,column) to &numcols;
               if denom = 0 then ps(i,j) = 0;
              else ps(i,j) = np(i,j)/denom;
            end;
          end;
end;
else if type = 11 then do;
          do j = columnl to columnr;
              denom = denom + np(row,j);
          end;
          do j = columnl to columnr;
```

```
            if denom = 0 then ps(row,j) = 0;
            else ps(row,j) = np(row,j)/denom;
         end;
end;

else if type = 12 then do;
         do i = rowl to rowr;
            do j = i- &dwod to column;
              denom = denom + np(i,j);
            end;
         end;
         do i = rowl to rowr;
            do j = i- &dwod to column;
              if denom = 0 then ps(i,j) = 0;
             else ps(i,j) = np(i,j)/denom;
            end;
         end;
end;

else if type = 13 then do;
         do i = rowl to rowr;
            do j = column to &numcols;
              denom = denom + np(i,j);
            end;
         end;
         do i = rowl to rowr;
            do j = column to &numcols;
              if denom = 0 then ps(i,j) =0;
             else ps(i,j) = np(i,j)/denom;
            end;
         end;
end;

else if type = 14 then do;
         do i = 1+ &dwod to row;
            do j = columnl to columnr;
               denom = denom + np(i,j);
            end;
         end;
         do i = 1+ &dwod to row;
            do j = columnl to columnr;
              if denom = 0 then ps(i,j) = 0;
```

```
                else ps(i,j) = np(i,j)/denom;
              end;
          end;
end;

else if type = 15 then do;
          do i = row to min(&numrows,columnr+ &dwod);
              do j = max(i- &dwod,columnl) to columnr;
                 denom = denom + np(i,j);
              end;
          end;
          do i = row to min(&numrows,columnr+ &dwod);
              do j = max(i- &dwod,columnl) to columnr;
                if denom = 0 then ps(i,j) = 0;
                else ps(i,j) = np(i,j)/denom;
              end;
          end;

end;

else if type = 16 then do;
          do i = rowl to rowr;
             do j = columnl to columnr;
                  denom = denom + np(i,j);
             end;
          end;
          do i = rowl to rowr;
             do j = columnl to columnr;
                 if denom = 0 then ps(i,j) = 0;
                 else  ps(i,j) = np(i,j)/denom;
             end;
          end;
end;

else if type = 17 then ps(1,column) = 1;

else if type = 18 then do;

           do j = 1 to column;
               denom = denom + np(1,j);
           end;
           do j = 1 to column;
```

```
                if denom = 0 then ps(1,j) = 0;
              else  ps(1,j) = np(1,j)/denom;
            end;
end;

else if type = 19 then do;
            do j = column to &numcols;
                  denom = denom + np(1,j);
            end;
            do j = column to &numcols;
                do i = min(&numrows,column+1) to min(&numrows,j+1);
                   denom = denom + np(i,j);
                end;
            end;
            do j = column to &numcols;
                 if denom = 0 then ps(1,j) = 0;
                 else ps(1,j) = np(1,j)/denom;
            end;
            do j = column to &numcols;
                do i = min(&numrows,column+1) to min(&numrows,j+1);
                   if denom = 0 then ps(i,j) = 0;
                  else ps(i,j) = np(i,j)/denom;
                end;
            end;
end;

else if type = 20 then do;
             do j = columnl to columnr;
                 denom = denom + np(1,j);
             end;
             do j = columnl to columnr;
                  do i = min(&numrows,columnl+1) to min(&numrows,j+1);
                     denom = denom + np(i,j);
                  end;
             end;
             do j = columnl to columnr;
                 if denom = 0 then ps(1,j) = 0;
               else  ps(1,j) = np(1,j)/denom;
             end;
             do j = columnl to columnr;
                  do i = min(&numrows,columnl+1) to min(&numrows,j+1);
                     if denom = 0 then ps(i,j) = 0;
```

```
                    else ps(i,j) = np(i,j)/denom;
                  end;
              end;
end;

else if type = 21 then do;
          if &dwod then    denom = np(1,column);
          do i = row to min(&numrows,column+ &dwod);
               denom = denom + np(i,column);
           end;
          if &dwod then do;
              if denom = 0 then ps(1,column) = 0;
           else                ps(1,column) = np(1,column)/denom;
           end;
          do i = row to min(&numrows,column+ &dwod);
            if denom = 0 then ps(i,column) = 0;
             else
               ps(i,column) = np(i,column)/denom;
          end;
end;

else if type = 22 then do;
       if &dwod then do;
          do j = 1 to column;
               denom = denom + np(1,j);
           end;
       end;

          do i = row to min(&numrows,column+ &dwod);
                do j = i- &dwod to column;

                  denom = denom + np(i,j);

                end;
            end;
         if &dwod then do;
            do j = 1 to column;
                if denom = 0 then ps(1,j) = 0;
              else  ps(1,j) = np(1,j)/denom;
            end;
          end;
           do i = row to min(&numrows,column+ &dwod);
```

```
                    do j = i- &dwod to column;
                     if denom = 0 then ps(i,j) = 0;
                     else ps(i,j) = np(i,j)/denom;
                    end;
               end;
end;

else if type = 23 then do;
       if &dwod then do;
          do j = column to &numcols;
                 denom = denom + np(1,j);
           end;
       end;
           do j = column to &numcols;
               do i = min(&numrows,column+&dwod) to min(&numrows,j+&dwod);
                  denom = denom + np(i,j);
               end;
           end;

          do i = row to min(&numrows-1,column +&dwod -1);
            do  j= column to &numcols;
              denom = denom + np(i,j);
            end;
          end;
         if &dwod then do;
          do j = column to &numcols;
                if denom = 0 then ps(1,j) = 0;
                else ps(1,j) = np(1,j)/denom;
           end;
          end;
           do j = column to &numcols;
               do i = min(&numrows,column+&dwod) to min(&numrows,j+&dwod);
                  if denom = 0 then ps(i,j) = 0;
                  else ps(i,j) = np(i,j)/denom;
               end;
           end;
           do i = row to min(&numrows-1,column +&dwod -1);
            do  j= column to &numcols;
               if denom = 0 then ps(i,j) = 0;
              else ps(i,j) = np(i,j)/denom;
            end;
          end;
```

```
end;

else if type = 24 then do;
          if &dwod then do;
             do j = columnl to columnr;
                denom = denom + np(1,j);
            end;
          end;
          do i = row to min(&numrows-1,columnl+ &dwod-1);
              do j = columnl to columnr;
                 denom = denom + np(i,j);
              end;
          end;

         do i = min(&numrows,columnl+ &dwod) to min(&numrows,columnr+ &dwod);
              do j = max(i- &dwod,columnl) to columnr;
                 denom = denom + np(i,j);
              end;
          end;
          if &dwod then do;
           do j = columnl to columnr;
                if denom = 0 then ps(1,j) = 0;
              else  ps(1,j) = np(1,j)/denom;
            end;
           end;
          do i = row to min(&numrows-1,columnr+ &dwod-1);
              do j = max(i- &dwod,columnl) to columnr;
                if denom = 0 then ps(i,j) = 0;
                else ps(i,j) = np(i,j)/denom;
              end;
          end;
        do i =  min(&numrows,columnl+ &dwod) to min(&numrows,columnr+ &dwod);
              do j = max(i- &dwod,columnl) to columnr;
                if denom = 0 then ps(i,j) = 0;
                else ps(i,j) = np(i,j)/denom;
              end;
          end;

end;
run;

data pseud;
```

```
set &emout end = last nobs=datan;
array pseu{&numrows,&numcols};
array ps{&numrows,&numcols};
do i= 1 to &numrows;
   do j = 1 to &numcols;
       pseu(i,j) + ps(i,j);
   end;
end;
if last then do;
      n = 0;
    do i = 1 to &numrows;
       do j = 1 to &numcols;
        n = n + pseu(i,j);
       end;
    end;

if (round(n) - round(datan)) then do;
     put 'Pseudo data does not add up';
     put 'n is ' n ' while datan is ' datan;
    end;
end;
call symput('emtot',left(trim(n)));
if last;
keep pseu1 - pseu&total;
run;

/****This data set saves last set of probs.****/
data oldprobs;
set nextprob;
array newp{&numrows,&numcols};
array oldp{&numrows,&numcols};
do i = 1 to &numrows;
  do j = 1 to &numcols;
       oldp(i,j) = newp(i,j);
  end;
end;
keep oldp1 - oldp&total;
run;

/****This data set generates the next set of probabilities.****/
data nextprob;
set pseud;
```

```
array pseu{&numrows,&numcols};
array newp{&numrows,&numcols};
do i = 1 to &numrows;
   do j = 1 to &numcols;
       newp(i,j) = pseu(i,j)/&emtot;
  end;
end;
keep newp1 - newp&total;
run;

/***These probs should then be fed into the first data step.
****But first we need to compare these probs to the last
****probs to check for convergence.***/
/****testing for convergence.*******/
data _null_;
merge oldprobs nextprob;
array newp(&numrows,&numcols);
array oldp(&numrows,&numcols);
diffsq = 0;
do i = 1 to &numrows;
  do j = 1 to &numcols;
      diffsq =diffsq + (newp(i,j) - oldp(i,j))* (newp(i,j) - oldp(i,j));
  end;
end;
call symput('diffsq',left(trim(diffsq)));
run;

data _null_;
a = &diffsq;
b = &tol;
/*put 'a is ' a;
put 'b is ' b;*/
if a < b then do;
   ind = 1;
   call symput('ind',left(trim(ind)));
end;
stop;
run;

/*%put ind is &ind; */
%let i = %eval(&i +1);
%let numits = %eval(&i-1);
```

```
%if &i >= %eval(&its) %then %do;
      %put We have reached &its iterations with sum squared differences
equal to &diffsq;
     %end;
%end;

/******For simulation put &numits into data set to get
******convergence rate.***/
%put The final numits is &numits;
%put The final diffsq is &diffsq;
data &probout;
set nextprob;
run;
%mend;
/* em(datafile,probfile,emout,probout,numrows,numcols,total,tol,its)*/
/*%em(outmm,newprobs,pseudo,finalprb,&numrows,&numcols,&total,.0001,10);*/
/*test to seee if total pseudo data adds to total in mortmorb*/
/*data _null_;
set pseud;
array pseu{&numrows,&numcols};
sum = 0;
do i= 1 to &numrows;
 do j = 1 to &numcols;
    sum = sum + pseu(i,j);
 end;
end;
call symput('pseutot',left(trim(sum)));
run;
%put pseudo count total seems to be &pseutot; */
/*Testing distribution sums by type.*/
/*proc sort data=&emout;
by type;
run;
data test;
set &emout;
by type;
array ps{&numrows,&numcols};
         sum=0;
          do i = 1 to &numrows;
             do j = 1 to &numcols;
```

```
            sum = sum + ps(i,j);
          end;
        end;
keep sum type;
run;
proc univariate data = test;
var sum;
by type;
run;*/
%macro estcov(matrix,estcov);
/*************************************************************
***This macro computes covariance matrix for a vector of   ***
***multinomial cell probabilities.                         ***
***Matrix is a data set containing the k rows and columns  ***
***of an information matrix in variables named col1 through***
***colk.  It also contains  variables named p, row and    ***
***column.  The variable named p contains                  ***
***the first k of k+1 probabilities.  The variables row and***
***column conain the row and column number of the original ***
***matrix of rectangles for the probability in p.  Letting ***
***im and p be the matrix and probability vector, the      ***
***output will be in the data set named in estcov.  The    ***
***data set,estcov, will have k observations with variables***
***cov1 through covk,row, col and p. Cov1 through covk will***
***contain the inverse of col1 though colk. Row, col and p ***
***will be the same as in matrix.  Cov1 through covk will  ***
***covariance matrix for the vector of multinomial         ***
***be the estimates, p.                                    ***
*************************************************************/

data _null_;
set &matrix nobs=n;
infdim = n;
call symput('infdim',compress(infdim));
stop;
run;

data _inf_;
set &matrix;
keep col1-col&infdim;
run;
```

```
data _rc_;
set &matrix;
keep row column p;
run;

proc iml;
/****converting matrix to matrix and vector form****/
 use _inf_;
   read all var _num_ into inf;

  cov = inv(inf);
create cov from cov;
  append from cov;
quit;

data &estcov;
merge cov _rc_;
array col{&infdim};
array cov{&infdim};
do i = 1 to &infdim;
   cov(i) = col(i);
end;
drop col1-col&infdim i;
run;
%mend;
%macro pstdtab(covp,rowfmt,colfmt,colen,fmtcell,title);
/*********************************************************
***This macro tables the joint probabilities and their***
***standard deviations for a two dimensional matrix of***
***of rectangles.                                       ***
***covp is a SAS data set containing k observations.  ***
***The k observations contain variables cov1-covk, row***
***col and p.  Cov1-covk is the covariance matrix for ***
***the multinomial probability estimates in p.  Row   ***
***and col contain the row and column for the values  ***
***in p in the original matrix of rectangles.  There  ***
***are k+1 probabilities.  The missing probability is ***
***the probability for the lower right rectangle of   ***
***matrix.  The row and column numbers for this       ***
***probability are max of the row values and max of   ***
***the column values, respectively.                    ***
***The last probability is computed as 1 - the sum of ***
```

```
***the values in p.  Its variance is the sum of the   ***
***entries in the covariance matrix, cov1-covk.       ***
***Title will be the title of the table.              ***
***Rowfmt and colfmt are the formats that for the rows***
***and column.  For example row =1 may be death w/o   ***
***disease and the rest of the rows and columns may   ***
***partitions of (0,infinity).                        ***
***colen is the width desired for the table columns.  ***
***fmtcell is the format in w.d form (e.g. 7.3) for   ***
***the values to be printed in the table cells.       ***
***Note that this macro includes                      ***
***c:\windows\desktop\mortmorb\tabmac.sas.            ***
*********************************************************/
%include 'c:\windows\desktop\mortmorb\tabmac.sas';
data _null_;
set &covp nobs = n;
covdim = n;
call symput('covdim',compress(covdim));
stop;
run;

/*****The following code computes the final probability ***
******and the standard deviations.                       ***
*********************************************************/

data stdprob_;
set &covp end=last;
array cov{&covdim};
lastvar + sum(of cov1-cov&covdim);
totprob + p;
i = _n_;
stdev = sqrt(cov(i));
prob  = p;
if last then do;
   lprob = 1.0 - totprob;
   lstdev = sqrt(lastvar);
   call symput('lprob',compress(lprob));
   call symput('lstdev',compress(lstdev));
end;
keep prob stdev;
run;
```

```
data lastones;
prob = &lprob;
stdev = &lstdev;
output;
stop;
run;

data stdprob_;
set stdprob_ lastones;
run;
/****Adding the row and column for the left out cell***
*****to the row column list.**/
data _rc_ ;
set &covp;
keep row column;
run;

data _lastrc_;
set _rc_ end = last;
retain maxj;
if _n_ = 1 then maxj = column;
else maxj = max(maxj,column);
if last then do;
    if column < maxj then column = column + 1;
    else row = row + 1;
end;
if last;
keep row column;
run;

data _rc_;
set _rc_ _lastrc_;
run;
/********Putting probs, stdev, row and column into data set***
*********suitable for use in the tableit macro.            **/
data _final_;
merge stdprob_ _rc_;
run;
%tableit(_final_,row,column,&rowfmt,&colfmt,&colen,&fmtcell,
                prob stdev,time-to-onset,time-to-death,&title);
%mend;
/****Example*******/
```

```
%include 'c:\windows\desktop\mortmorb\matrx.sas';
%matrx(mm.rinform,rmats);
/*%matrx(mm.cinform,cmats);*/
/* estcov(matrix,estcov)
 pstdtab(covp,rowfmt,colfmt,colen,fmtcell,title)*/

/*%estcov(rmats,rcov);
options notes;
%pstdtab(rcov,rrow,rcolumn,10,8.6,Exposed Joint Probabilities); */
%macro fillcov(covp,filled);

/*********************************************
***This macro takes the covariance matrix   ***
***for a vector of K probabilities of a K+1 ***
***multinomial and fills in the K+1 row and ***
***column for the K+1 probability.  It also ***
***calculates the K+1 probability.          ***
***covp in a SAS input data set containing  ***
***the covariance matrix in K observatons   ***
***and variables cov1 through covK and the K***
***probabilities in the variable p.         ***
***Variables Row and colomn will contain    ***
***the coordinates for the rectangle for    ***
***which the p is the probability.          ***
***Filled is a SAS output data set with K+1 ***
***observations  and variable cov1-cov(K+1) ***
***containing the covariance matrix for the ***
***vector of K+1 probabilities in variable p***
***Row and colomn will be updated to include***
***the coordinates of the rectangle for the ***
***last probability.                        ***
*********************************************/
/*computing last row and column numbers*/

data _lastrc_;
set &covp end = last;
retain maxj;
if _n_ = 1 then maxj = column;
else maxj = max(maxj,column);
if last then do;
    if column < maxj then column = column + 1;
    else row = row + 1;
```

```
end;
if last;
keep row column;
run;

data _null_;
set &covp nobs=n;
call symput('covdim',compress(n));
stop;
run;

data lastrow;
set &covp end=last;
array _sum{&covdim};
array cov{&covdim};
do i = 1 to &covdim;
    _sum(i) + cov(i);     /*summing column i of cov matrix.*/
end;
sump + p;                /*summing the probabilities*/
if last then do;
    do i=1 to &covdim;
       cov(i) = -_sum(i); /*computing K+1 observation for cov1-covK*/
    end;
       p = 1.0 - sump;  /*computing K+1 probability.  */
end;
if last;
keep cov1-cov&covdim p;
run;

%let filldim = %eval(&covdim + 1);

data lastcol;
set lastrow;
array cov{&covdim};
retain cov1-cov&covdim; /*puts sum of columns of original cov matrix*/
do i = 1 to &covdim;    /*into a column under the name cov&filldim.*/
       cov&filldim = cov(i);
       output;
end;
drop i p cov1-cov&covdim;
run;
```

```
data lastrow;
set lastrow;
array cov{&filldim};
cov(&filldim) = 0;
do i = 1 to &covdim;
     cov(&filldim) = cov(&filldim) - cov(i);
/*summing all original cov entries. This is the variance of the estimate for
the K+1 probability.*/
end;
drop i;
run;

data lastrow;
merge lastrow _lastrc_;
run;

data &filled;
merge &covp lastcol;
run;

data &filled;
set &filled lastrow;
run;
%mend;
/*example*/
/*data test;
input cov1-cov7 p row column;
cards;
   1 2 3 4 5 6 7 .1 1 1
   2 3 4 5 6 7 1 .1 1 2
   3 4 5 6 7 1 2 .1 1 3
   4 5 6 7 1 2 3 .1 2 1
   5 6 7 1 2 3 4 .1 2 2
   6 7 1 2 3 4 5 .1 2 3
   7 1 2 3 4 5 6 .1 3 2
;
run;

%fillcov(test,outest);
proc print data = outest noobs;
run;*/
 /***Fixboth.sas*****/
```

```
/***This code runs mm2 on Exposed and control data. Feeding output***
****formats from Exposed run into the run for comparisons.**/
options ls=78 nonotes;
libname mm 'c:\windows\desktop\mortmorb';
%include 'c:\windows\desktop\mortmorb\mm2.sas';
%mm2(mm.exposed,mm.rprobs,mm.rpseu,dis,deth,times,.0001,100,
                        c:\windows\desktop\mortmorb,mm.rinform,r);
%mm2(mm.control,mm.cprobs,mm.cpseu,rdis,rdeth,times,
               .0001,100,c:\windows\desktop\mortmorb,mm.cinform ,c);
/*%mortmorb(mm.control,mm.cprobs,mm.cpseu,20,.0001,100,
                        c:\windows\desktop\mortmorb,mm.cinform,c); */
/****intprob2.sas   3/30/98**/
%macro initprob(infile, outfile);
/*********************************************************
***This macro takes an infile which is the output from a***
***proc freq run and produces an outfile which is a set ***
***of probabilities for each cell of the table.         ***
***The table created is such that removing the first row***
***leaves a submatrix which haszero counts for the      ***
***triangle below the diagonal.                         ***
***Cells outside of the lower triangle that have zero   ***
***counts are given the minimum adjusted count of 1/n or***
***sum/n where n is the total of the table and sum      ***
***is the sum of the actual counts of all cells adjacent***
***to the empty cell.  The outfile of probabilities will***
***be the adjusted count for a cell divided by the sum  ***
***of the adjusted counts for all the cells.            ***
***The order of the cells is that of the order of a two ***
***dimensional array in SAS.  That is, cells are counted***
***across columns and then down to the next row.        ***
***The number of columns, rows and the total number of  ***
***cells are put into global macro variables named      ***
***numcols, numrows and total, respectively.            ***
*********************************************************/
%global numcols numrows total dwod;
/**dwod indicates whether or not the first row of the matrix ***
***is death without disease.  If dwod is 1 then there is such***
***a row if dwod is 0 then there is not.                    **/
/**Next data step put the appropriate values into the global macros.***/
data _null_;
set &infile nobs = n;
by obst;
```

```
if _n_ = 1 then do;
              numcols = 1;
    if obst = . then do;
   dwod = 1;

    end;
    else do;
        dwod = 0;
    end;
   call symput('dwod',compress(dwod));  /*dwod is 1 if there is a death
                                        **without disease row and zero
                                        **otherwise.*/
end;
else numcols = numcols + 1;
retain numcols;
if last.obst then do;
    numrows = n/numcols;
    call symput('numcols',left(trim(numcols)));
    call symput('numrows',left(trim(numrows)));
    call symput('total',  left(trim(n)));
    stop;
end;
run;
/***The following data set puts the cell index (i,j) in the***
****data.  Each observation has then a cell count and the  ***
****cell index.                                            **/
data cells;
set &infile end = last;
by obst;
i = _n_/&numcols + 1;
i = int(i);
if last.obst then i = i-1;
if first.obst then j = 1;
else j = j+1;
retain j;
run;
/****We will not remove impossible cells.  We will just keep them
***at count = 0 and prob = 0.  It makes the indexing easier,
***although more inefficient. **/
/**The next data step takes the counts for each cell and computes**
***the sample proportion in each cell, putting the sample        **
***proportion into an array named probs with the appropriate cell**
```

```
***index (i,j).                                                    */
data probs;
set cells end = last;
by obst;
array cnts{&numrows,&numcols};
array probs{&numrows,&numcols};

retain  cnts1-cnts&total
        probs1-probs&total;
cnts(i,j) = count;
probs(i,j)  = percent/100;
sum + count;
if last then do;
     call symput('n',left(trim(sum)));
             end;
if last;
drop  count percent obst obsd;
run;
/*********************************************************************
***Keep in mind that the two dimensional arrays are numbered     ***
***in the following way (i,j) is (i-1)*numcols+int(j/numcols + 1)***
***for jmod(numcols) ne 0 and i*numcols if jmod(numcols) is 0.   ***
*********************************************************************/
/*********************************************************************
***Now we must adjust the probs array in data set probs so that  ***
***0 count cells get small probabilities in accordance to the    ***
***counts in adjacent cells.                                     ***
***Keep in mind that we will keep the count at 0 for any cell,   ***
***(i,j) where i-&dwod > j, because these are cells in which     ***
***diseasetime is greater than death time.                       ***
*********************************************************************/
data &outfile;
set probs;
array cnts{&numrows,&numcols};
array probs{&numrows,&numcols};
array newp{&numrows,&numcols};
if cnts(1,1) = 0 then
     newp(1,1) = cnts(1,2) + cnts(2,1) + cnts(2,2);
do jj = 2 to %eval(&numcols -1);
     if cnts(1,jj) = 0 then do;
          newp(1,jj) = 0;
         do ii = 1 to 2;
```

```
                do jjj = jj-1 to jj+1;
                 newp(1,jj) = newp(1,jj) + cnts(ii,jjj);
                end;
                              end;
           end;
end;
if cnts(1,&numcols) = 0 then do;
          newp(1,&numcols) = 0;
     do jj = %eval(&numcols -1) to &numcols;
          newp(1,&numcols) = newp(1,&numcols) + cnts(2,jj);
     end;
                               end;   /****row 1 is now taken care of.**/

do ii = 2 to %eval(&numrows - 1);
    do jj = 2 to %eval(&numcols - 1);
           if cnts(ii,jj) = 0 then do;
              newp(ii,jj) = 0;
                 do iii = ii-1 to ii+1;
                   do jjj= jj-1 to jj+1;
                     newp(ii,jj) = newp(ii,jj) + cnts(iii,jjj);
                   end;
                 end;
                                      end;
    end;
end;           /*This takes care of all cells in interior of table.**/

do ii = 2 to %eval(&numrows -1);
       if cnts(ii,1) = 0 then do;
             newp(ii,1) = 0;
            do iii=ii-1 to ii+1;
               do jjj = 1 to 2;
                 newp(ii,1) = newp(ii,1) + cnts(iii,jjj);
               end;
            end;
                               end;
end;               /*This takes care of the first column up to second
                     last row.*/
if cnts(&numrows,1) = 0 then do;
          newp(&numrows,1) = 0;
             do iii = %eval(&numrows -1 ) to &numrows;
               do jjj = 1 to 2;
```

```
                newp(&numrows,1) = newp(&numrows,1) + cnts(iii,jjj);
              end;
            end;
                              end;/*This takes care of column 1 of last row.*/

do ii = 2 to %eval(&numrows - 1);
          if cnts(ii,&numcols) = 0 then do;
               newp(ii,&numcols) = 0;
            do iii = ii-1 to ii+1;
              do jjj = %eval(&numcols - 1) to &numcols;
               newp(ii,&numcols) = newp(ii,&numcols) + cnts(iii,jjj);
              end;
            end;
                                          end;
end;                  /**This takes care of last column up to last row*/

if cnts(&numrows,&numcols) = 0 then do;
            newp(&numrows,&numcols) = 0;
              do iii = %eval(&numrows - 1) to &numrows;
                 do jjj = %eval(&numcols - 1) to &numcols;
                   newp(&numrows,&numcols) =
                                newp(&numrows,&numcols) + cnts(iii,jjj);
                 end;
              end;
                                       end;
do jj=2 to %eval(&numcols - 1);
            if cnts(&numrows,jj) = 0 then do;
              newp(&numrows,jj) = 0;
              do iii= %eval(&numrows -1) to &numrows;
                 do jjj = jj-1 to jj+1;
                 newp(&numrows,jj) = newp(&numrows,jj) + cnts(iii,jjj);
                 end;
              end;
                                              end;
end;      /**This takes care of last row cols 2 to second last col.*/
             /***Now all 0 cells should be done.**/

do jj = 1 to &numcols;
     if cnts(1,jj) ne 0 then newp(1,jj) = cnts(1,jj);
      if newp(1,jj) = 0 then newp(1,jj) = 1.0/sum;

   end;
```

```
do ii = 2 to &numrows;
   do jj = ii-1 to &numcols;
      if cnts(ii,jj) ne 0 then newp(ii,jj) = cnts(ii,jj);
      if newp(ii,jj) = 0 then newp(ii,jj) = 1.0/sum;

   end;
end;   /**Now new p contains either cell count if it is nonzero,
          and a small num that is at least 1/n and
          at most 1**/
/***The following sets the probs of impossible cells back to 0.***
****Notice that if dwob is 1 then there is a first row with    ***
****disease missing and thus the impossible cells start below  ***
****row 3.  If dwob is 0 then the entire matrix has lower      ***
****diagonal of impossible cells.                              **/


do ii = 2+&dwod to &numrows;
    do jj = 1 to ii-1-&dwod;
       newp(ii,jj) = 0;
    end;
end;
newsum=0;
do ii= 1 to &numrows;
     do jj= 1 to &numcols;
         newsum = newsum + newp(ii,jj);
     end;
end;

do ii= 1 to &numrows;
     do jj=1 to &numcols;
         newp(ii,jj) = newp(ii,jj)/newsum;
     end;
end;   /**Now newp contains adjusted probabilities**/
keep newp1-newp&total;
run;

%mend;

/*%initprob(newprobs);*/
%macro lethalty(covp,outfile,colfmt,title,collab);

/************************************************************
```

```
***This macro computes the of a disease for a discrete set ***
***of time intervals, using the joint distribution of      ***
***time-to-onset of disease and time-to-death.  It assumes ***
***that time-to-onset and time-to-death have been          ***
***partitioned into intervals and a matrix of rectangles   ***
***have been formed by these intervals.  Time-to-onset is  ***
***represented by the rows of the matrix, the first of     ***
***is a row for death known to have occurred before without***
***disease.  Time-to-death intervals form the columns of   ***
***matrix of rectangles.  The row and column intervals must***
***be the same although there may be more intervals for    ***
***time to death than for time-to-onset.  So the last      ***
***interval for time-to-onset will have a left endpoint    ***
***which coincides with one of the left endpoints for      ***
***time-to-death, but may be a collapsing of a number of   ***
***last intervals for time-to-death.                       ***
***The name of the input SAS data set in put into argument ***
***covp.                                                   ***
***Covp has K observations where K is the number of        ***
***rectangles in the matrix that have nonzero probability. ***
***Covp has variables cov1-covK, which are the columns for ***
***the covariance matrix of the K values in the variable,  ***
***p.  The variable,p contains the K estimates of the joint***
***probabilities for the rectangles.  The ordering of the  ***
***rectangles and thus the joint probabilities is the same ***
***as the conventional double array ordering in SAS.  That ***
***is the count start at the first row first column and     ***
***procedes right through the columns and then bac down to ***
***the next row.  The submatrix left after eliminating the ***
***first row has all zero joint probabilities in the lower ***
***triangle, since these rectangles represent knowing that ***
***disease occured after death and knowing exactly what    ***
***time interval the disease occured.  These rectangle are ***
***not included in covp.  Covp also contains variables row ***
***and column which indicate the coordinates of the joint  ***
***probability in the matrix of rectangles.                ***
***In matrix/vector form the K observations for cov1-covK  ***
***represent the covariance matrix for the K observations  ***
***in the column vector p.
***Lethality is computed for each time-to-death interval.  ***
***It is the sum of the joint probabilities for rows 2 to  ***
***the maximum time-to-onset interval less than or equal to***
```

```
***the death interval divided by the joint probability of  ***
***the first row (death w/o disease) in that death interval***
***In other words, the lethality for interval j is the     ***
***sum of the probabities of getting the disease up to and ***
***including interval j and dying in interval j divided by ***
***the probability of dying in interval j without disease. ***
***Outfile will be a SAS data set that contains the        ***
***the lethalities in a variable L, and the covariance of  ***
***the lethalities in columns Lcov1-LcovJ, where J is the ***
***number of time-to-death intervals(i.e. the number of    ***
***columns).  Colfmt is the format for assinging the column***
***intervals to the column numbers.                        ***
***Title is used to identify the data set.  It will be     ***
***the third line in the output print out title.  The first***
***two lines of the title are 'Column Lethalities and      ***
***Covariance Matix' and 'for' .  Collab is the lable to be***
***given to the column--e.g. time-to-death.                ***
***The covariance matrix for the vector of lethalities is  ***
***computed using the delta method.  The vector of         ***
***lethalities is a function of the p vector from the input***
***SAS data set covp.  So the asymptotic covariance matrix ***
***for the vector of lethalities is a product of the       ***
***gradient matrix of the Lethality vector times the matrix***
***in cov1-covK times the transpose of the gradient matrix.***
************************************************************/
/************************************************************
***The following data steps compute the numerators of the  ***
***lethalities and stores them with the denominators in a  ***
***data set _nums_.                                        ***
************************************************************/
data _null_;
set &covp nobs = K end = last;
retain maxrow maxcol;
if _n_ = 1 then do;
     maxrow = row;
     maxcol = column;
end;
else do;
     maxrow = max(maxrow,row);
     maxcol = max(maxcol,column);
end;
if last then do;
```

```
      call symput('K',compress(K));
      call symput('rownum',compress(maxrow));
      call symput('colnum',compress(maxcol));
end;
run;

data _in_;
set &covp end=last;
array num{&colnum}; /**For numerator of column lethalities.*/
array den{&colnum}; /**For denominator of column lethalities.*/
retain num1-num&colnum den1-den&colnum;
if _n_ = 1 then do;
      do j = 1 to &colnum;
          num(j) = 0;

      end;
end;
           if row = 1 then den(column) = p;
         else num(column) = num(column) + p;

if last;
keep  num1-num&colnum den1-den&colnum;
run;
/*****************************************************************
***The following data set computes the gradient matrix for the ***
***lethality vector.  The gradient matrix has colnum rows and  ***
***K columns.                                                  ***
***Most columns for a row are zero.  The jth. row has a nonzero***
***entrie in column j, which is, in the variables of _in_,     ***
***-numj/(denj*denj).  The other nonzero entries are 1/denj and***
***appear in columns corresponding to the matrix coordinates   ***
***(min(2,rownum),j) to (min(j+1,rownum),j).                   ***
***The column corresponding to coordinates (i,j) is            ***
***s = (i-1)*colnum + j -(i-2)*(i-1)/2.                         ***
*****************************************************************/
data _prime_;
set _in_;
array num{&colnum};
array den{&colnum};
retain num1-num&colnum den1-den&colnum;
do j = 1 to &colnum;
       sum = num(j);
```

```
       p1  = den(j);
       output;
end;
drop num1-num&colnum den1-den&colnum;
run;

data fprime;
set _prime_;
array f{&k};
do j = 1 to &k;
     f(j) = 0;
end;
f(_n_) = -sum/(p1*p1);
do i = min(2,&rownum) to min(_n_ + 1,&rownum);
     s = (i-1)*&colnum + _n_ - (i-2)*(i-1)/2;
     f(s) = 1.0/p1;
end;
drop i j s;
run;

data fprime;
set fprime;
leth = sum/p1;
drop sum p1;
run;

/**preparing matrices for use in IML**/

data _cov_;
set &covp;
keep cov1-cov&k;
run;

data _temp_;
set fprime;
keep f1-f&k;
run;

proc iml;
/***putting the fprime matrix into a matrix and lethality vector***
****into a vector.                                                **/
use _temp_;
```

```
   read all var _num_ into _fprime_;
use _cov_;

    read all var _num_ into _covp_;

    covleth = _fprime_*_covp_*_fprime_';
   create newcov from covleth;
   append from covleth;
quit;

data newcov;
set newcov;
array col{&colnum};
array lcov{&colnum};
do i = 1 to &colnum;
      lcov(i) = col(i);
end;
keep lcov1-lcov&colnum;
run;

data _leth_;
set fprime;
column = _n_;
keep leth column;
run;

data &outfile;
lable leth ='lethality' column = &collab;
merge _leth_ newcov;
run;

proc print data = &outfile label;
title1 'Column Lethalities and Covariance Matix';
title2 'for';
title3 &title;
format column &colfmt..;
var column leth lcov1-lcov&colnum;
run;
%mend;
/*
%lethalty(rfilld,rout,rcolumn,Exposed,Time-to-Death);
```

```
%lethalty(cfilld,cout,rcolumn,Controls,Time-to-Death);  */
/********lethout.sas****/
/****This code computes and tables the lethalities for Exposed ***
*****and comparisons by time-to-death interval and tests for equality***
*****of the two vectors of lethality.                              ***
**************************************************************************/

%include 'c:\windows\desktop\mortmorb\lth2quad.sas';
%include 'c:\windows\desktop\mortmorb\quaddiff.sas';
%include 'c:\windows\desktop\mortmorb\lethalty.sas';
     %lethalty(rfilld,rout,rcolumn,Exposed,Time-to-Death);
     %lethalty(cfilld,cout,rcolumn,Controls,Time-to-Death);
     %lth2quad(rout,inr);
     %lth2quad(cout,inc);
     %quaddiff(inr,inc,rrow,rcolumn,lethality,Exposed,Controls);
%macro lth2quad(lethin,lethout);

/*********************************************
***This macro takes the output from lethalty***
***and puts it into the form for input to   ***
***quaddiff.                                ***
***Input data set lethin is output from     ***
***lethalty and lethout will be the output  ***
***data set suitable for immediate input to ***
***diffquad.                                ***
*********************************************/

data _null_;
set &lethin nobs = n;
call symput('n',compress(n));
stop;
run;

data &lethout;
set &lethin;
array lcov{&n};
array cov{&n};
do i = 1 to &n;
      cov(i) = lcov(i);
end;
p = leth;
row = 0;
```

```
keep p row column cov1-cov&n;
run;
%mend;
/**example**/
 /*
%include 'c:\windows\desktop\mortmorb\lth2quad.sas';
%include 'c:\windows\desktop\mortmorb\quaddiff.sas';
%include 'c:\windows\desktop\mortmorb\lethalty.sas';
     %lethalty(rfilld,rout,rcolumn,Exposed,Time-to-Death);
     %lethalty(cfilld,cout,rcolumn,Controls,Time-to-Death);
     %lth2quad(rout,inr);
     %lth2quad(cout,inc);
     %quaddiff(inr,inc,rcolumn,lethality,Exposed,Controls); */
/**********This code takes a data set containing
arrays im(infdim,infdim), newp(infdim+1), ni(infdim+1) and nj(infdim+1)
and puts them into
row/column matrix form with im(i,1)-im(i,infdim) the ith. row where
cols are named col1-colinfdim.  newp(i) is the value of a variable
name p in the ith observation.  Row = ni(i) and column = nj(i) are the
row and column rectangle for which p is the probability.  The last newp,
ni and nj values are not included since the last p value is 1 minus the
sum of the first infdim p values, ni(infdim + 1) = ni(infdim) and
nj(infdim + 1) = nj(infdim) + 1, assuming that there is a row for disease
known not to have occurred before death.
**********/

/**remember that there is one more probabilitiy the last probability is
1 minus the sum of the infdim p's.***/

%macro matrx(in,out);

data _null_;
set &in;
newpsize = infdim + 1;
call symput('newpsize',compress(newpsize));
call symput('infdim',compress(infdim));
stop;
run;

data &out;
set &in;
array im{&infdim,&infdim};
```

```
array col{&infdim};
array newp{&newpsize};
array ni{&newpsize};  /**new**/
array nj{&newpsize};  /**new**/
do i = 1 to &infdim;
    do j= 1 to &infdim;
        col(j) = im(i,j);
    end;
  p = newp(i);
  row = ni(i);   /*new*/
  column = nj(i); /*new*/
  output;
end;
keep col1-col&infdim p row column; /*new*/
run;
%mend;
/*matrx(in,out)*/
%matrx(mm.rinform,rmats);
/*****mm2.sas******/
/*******************Mortality Morbidity Analysis Software.**********/
/*************************************************************************
***The main driver macro named mortmorb requires a data file that  ***
***has mortality morbidity data in the following form.              ***
***Time to disease and time to death each have five variables       ***
***associated with them.  Two of the variables are censoring       ***
***indicators and the other three are the available information     ***
***about the time to event (disease or death).                     ***
***The three variables that indicate the known value of disease are***
***obst, obstl and obstr.  Obstl and obstr are the left and right  ***
***interval endpoints if time to disease is doubly censored.  That ***
***is, if time to disease is known only to fall into an interval   ***
***then that interval is (obstl,obstr).                            ***
***In this case obst, which is the observed value of time to       ***
***disease is '.' , because it is unobserved and considered missing.**
***If time to disease is observed or singly censored then the value***
***or the censored value is in obst, while obstl and obstr will be ***
***'.' .   The censoring indicators for time to disease must be    ***
***named itL and itR.  ItL = 1 if time to disease is left censored.***
***That is if max(time to disease,lower censoring variable) =      ***
***lower censoring value.  In this case we observe the value of the***
***censoring variable.  ItL = 0 if the maximum is equal to time to ***
***disease. ItR = 1 if time to disease is right censored.  That is ***
```

```
***if min(time to disease,upper censoring variable) = censoring    ***
***variable.  In this case we observe the value of the censoring   ***
***variable.  ItR = 0 if the min above is the time to disease.     ***
***If ItL = 0 and ItR =0 then obst is the actual time to disease.  ***
***If ItL = 1 and ItR =0 then the actual time to disease is less   ***
***than obst.                                                      ***
***If ItL = 0 and ItR =1 then the actual time to disease is greater***
***than obst.                                                     ***
***In all of the cases above obstL and obstR will be '.' because   ***
***none of these cases is double censored.                        ***
***If ItL=1 and ItR = 1 then obst = '.' and obstl <= actual disease***
***time <= obstr.                                                 ***
***If death occured before disease then obst, obstl and obstr are  ***
***all '.' and Itl = 0 and ItR =1.  This indicator pattern is used ***
***because technically death has censored disease time of the right.**
***The variables related to time to death are IdL,IdR, obsd, obsdL ***
***and obsdR.  They are defined similarly to those for disease.    ***
*********************************************************************/
%global numtimes;  /*contains number of disease and death times
                     to be used to form the lattice for disease
                     time by death time.*/
%include 'c:\windows\desktop\mortmorb\cellsgn2.sas';
%include 'c:\windows\desktop\mortmorb\classify.sas';
%include 'c:\windows\desktop\mortmorb\em.sas';
%include 'c:\windows\desktop\mortmorb\intprob2.sas';
%include 'c:\windows\desktop\mortmorb\time5two.sas';
%include 'c:\windows\desktop\mortmorb\realfreq.sas';
%include 'c:\windows\desktop\mortmorb\revzero.sas';
%include 'c:\windows\desktop\mortmorb\newinfrm.sas';
%macro mm2(datafile,outprobs,pseudo,disfmt,dethfmt,intimes,
              tol,its,libdrive,inform ,pref);
/*********************************************************************
***Datafile is a mortality morbidity file with itL, itr as the left ***
***and right censoring indicators for time to disease, obst as the  ***
***observed time to disease, which is either the actual time or the ***
***right or left censoring time depending on the censorship.  If ItL***
***and itR are both 1, indicating double censoring then obst is '.' ***
***and obstl, obstr contain the left and right endpoints of the     ***
***observed interval for time to disease.  If obst is '.' because   ***
***death occurred before disease then itl = 0 and itr = 1.  IdL, idr***
***obsd, obsdL, obsdr are defined similarly for the time to death,  ***
***except, of course, that death will not be cesored by disease, so ***
```

```
***death time should never be missing.                                    ***
***outprobs is the data set containing the final em algorithm             ***
***probability estimates for the cells defined by the formats disfmt,***
***for rows and dethfmt for column, saved in library.                      ***
***These formats were presumable built on control set of                ***
***mortality/morbidity data using the macro mortmorb.                     ***
***pseudo is the data set containing the final pseudo data count for***
***the  cells defined by diint and deint.                                  ***
***Intimes is the data set containing t1 - t&numtimes which are the ***
***times used for the column intervals.  The row intervals are made ***
***up of endpoints which are a subset of intimes.                          ***
***Tol is the value for the sum of differences squared between             ***
***the successive probability steps of the em algorithm that will   ***
***stop algorithm if the sum of squared differences is less than it.***
***its is the maximum number of iterations of the em algorthm that  ***
***is allowed.                                                             ***
***The information matrix for the resulting set of probabilities is ***
***in a data set named inform in an array named im of                      ***
***square dimension infdim*infdim where infdim is the number of     ***
***nonzero cell probabilities (the upper diagonal) minus 1.         ***
***Inform also contains the final probability estimates in an array ***
***named newp1-newp(infdim).  In addition inform contains variables ***
***numcols, numrows, and infdim which are the number of columns,    ***
***rows and the dimension of the information matrix.  Finally inform***
***contains arrays Ni1-Ni(infdim) and Nj1-Nj(infdim), which contain ***
***the indices in the matrix of rectangles for the newp array.      ***
***pref is one character prefix to be given to data sets final      ***
***formats for death and time to onset of disease.  The format for  ***
***the disease intervals will be &pref.dis and for death &pref.deth.***
*************************************************************************/
%realfreq(&datafile,freqout,&disfmt, &dethfmt, &libdrive);
%initprob(freqout,initprob); /**freqout the outfile from realfreq.
                          initprob is a set of initial probabilities for
                          the em cells.**/  /*in intprob2*/
%cellsgn(&intimes,&datafile,outmm,&numrows,&numcols,&dwod); /* outmm
                                    is the output which is the mortmorb
                                    file with observed and censoring
                                    observations classified by row if
                                    disease and column if death.  Numrow
                                    and numcol are &numrows and &numcols
                                    which are created by initprob.*/
                                    /*in cellsgn2.sas*/
```

```
%classify(outmm,outmm,&dwod);
%em(outmm,initprob,&pseudo,&outprobs,&numrows,&numcols,&total,&tol,&its
                                                              ,&dwod);
                                /* em is in em.sas*/
%let numtimes = %eval(&numcols - 1);
%nextzero(&outprobs,&intimes,&numrows,&numcols,&numtimes,newp,&outprobs,times);
                                /*next zero is in revzero.sas*/
%if %eval(&yes) = 1 %then %do;    /*if outprobs and times have changed
                                  **then we rerun the em process with the
                                  **new matrix of probs and array of times.*/
%put Warning Rectangles have been combined;
%let numrows = &newrows;
%let numcols = &newcols;
%let total   = &newtot;
%cellsgn(times,&datafile,outmm,&numrows,&numcols,&dwod); /*times is the
                                        file of
                                        timesfrom eminvals. datafile is the
                                        mortmorb file. outmm
                                        is the output which is the mortmorb
                                        file with observed and censoring
                                        observations classified by row if
                                        disease and column if death.  Numrow
                                        and numcol are &numrows and &numcols
                                        which are created by initprob.*/
%classify(outmm,outmm,&dwod);
%let numtimes = &newtime;
%em(outmm,&outprobs,&pseudo,&outprobs,&numrows,&numcols,&total,&tol,&its,
                                                            &dwod);
%end;
%ddformat(times,&numtimes,&numrows,&numcols,&pref);
                      /**This was moved from within the above loop
                      to accomodate a change in times from the proc freq
                      as well as from the nextzero**/
data &pref.times;
set times;
run;
%eminfo(&pseudo,&numrows,&numcols,&inform);/*in newinfrm.sas*/
%mend;
/*mortmorb(datafile,outprobs,pseudo,intobs,tol,its,libdrive,pref)*/
/*%mortmorb(mm.cancer,fnlprobs,fnlpseu,20,.0001,100,
                         c:\windows\desktop\mortmorb,inform,p);*/
 /**newinform.sas    3/30/98***/
```

```
%macro eminfo(pseudata,nrows,ncols,inform);
/************************************************
***This macro takes final em contributions for   ***
***a set of rectangles defined by a matrix with  ***
***nrows intervals for the vertical axis       ***
***and ncols intervals for the horizontal axis.***
***The vertical axis corresponds to time to     ***
***onset of disease.  The first row is the event ***
***of no disease onset before death.  Technically***
***then disease onset time is known only to have ***
***occured after death.  The columns are         ***
***intervals for death times.  The data can be   ***
***looked at as pseudo counts for cells of a     ***
***matrix with rows being disease onset times and***
***columns being death times.                    ***
***These pseudo counts are expected values for   ***
***the actual cell in which the observation falls***
***given the partial information.                ***
***For each observations the cell counts must    ***
***add to one.                                   ***
***Using Loius(1982) these pseudo counts are used***
***to estimate the information matrix for the    ***
***MLE's of the the probabilities of the cells,  ***
***which were obtained from the em algorithm.    ***
***Pseudata also contains the final estimates of ***
***the cell probabilities.  The pseudo data must ***
***be in a SAS array ps(&nrows,&ncols).      ***
***The final probabilities must be in a SAS array***
***np((&nrows,&ncols).  The information      ***
***matrix will be in an array called             ***
***im((s,s) where s is the number of possible    ***
***nonzero rectangles minus 1.  The minus 1      ***
***reflects the fact that the                    ***
***probabilities must add to 1.   The information***
***matrix, the vector of nonzero probabilities   ***
***and the number of rows and column in the      ***
***disease by time matrix and the number of      ***
***nonzero probabilities will be in the data set ***
***inform in im(infdim,infdim),nps(infdim+1) and ***
***numrows numcols and infdim.                   ***
***The data set inform will also contains two    ***
***other arrays ni{infdim+1} and nj{infdim+1}.   ***
```

```
***ni(s) and nj(s) will the the disease and death***
***intervals for the original matrix for the     ***
***newp(s) the sth. rectangle probability in the ***
***upper diagonal matrix.  newp(s) appears in the***
***inform data set also.                         ***
***For example, letting invim{infdim,infdim} be  ***
***the inverse of the information matrix, im, the***
***entry invim(s,t) is the estimate of the       ***
***covariance of newp(s) and newp(t), which      ***
***in terms of the original matrix of rectangles ***
***is the covariance between np(ni(s),nj(s)) and ***
***np(ni(t),nj(t)).                              ***
*************************************************/
/*************************************************
***The null data step sets the total number of   ***
***nonzero disease time by death time cells.     ***
***The first row, which is the row of death times***
***for death before disease has no impossible    ***
***cells.  The matrix formed by the disease by   ***
***death intervals minust the first row is upper ***
***triangular in the sense that the cells below  ***
***the diagonal correspond to death before       ***
***disease with disease time known--this can not ***
***happen so these cells have zero probability.  ***
***The macro variable total below is the number  ***
***of cells in the first row plus the number of  ***
***upperdiagonal and diagonal cells for the rest ***
***of the disease by death matrix.               ***
***Since the cell probabilities must add to 1 the***
***dimension of the vector of cell probabilities ***
***is total -1, which is put into the macro      ***
***variable infdim.  The information matrix for  ***
***the cell probabilities is the square of infdim***
***which is put into the macro variable infsize. ***
*************************************************/

data _null_;
total = &nrows*&ncols - (&nrows-2)*(&nrows -1)/2.0 ;
infdim = total - 1;
infsize =infdim*infdim; /**number of entries in info mat*/
call symput('total',left(trim(total)));
call symput('infdim',left(trim(infdim)));
```

```
call symput('infsize',left(trim(infsize)));
stop;
run;

/***The arrays ni and nj hold the row and column numbers for the**
****vector of the upper diagonal pseudo-data and probabilities. **
****These matrices are stretched out in the code to allow the   **
****Louis method to be used.  Ni and nj will allow a map back to**
****the nonzero cells of the disease death matrix.              **
*****************************************************************/
data rowcol;
array ni{&total}; /*holds the row for the stretched out matrix*/
array nj{&total}; /*holds the collum for the stretched matrxi*/
capj = &ncols;
capi = &nrows -1;
do jj = 1 to capj;        /*This do block puts the death before disease*/

    ni(jj) = 1;
    nj(jj) = jj;

end;
                          /*This do block assigns the upper triangular*/
do i = 1 to capi;         /*portion of the possible cells.            */
   do j= i to capj;
       s = i*capj - ((i-1)*i/2.0) +j;

      ni(s) = i+1;
      nj(s) = j;

   end;
end;
output;
stop;
drop jj i j s capi capj;
run;

data newpseud;
set &pseudata;
array np(&nrows,&ncols); /*contains cell probs including zeros.*/
array ps(&nrows,&ncols); /*contributions to each cell.*/
array nps{&total}; /*vector of psuedodata for the possible cells*/
array newp{&total};/*vector of cell probs for the possible cells*/
```

```
capj = &ncols;
capi = &nrows -1;
do jj = 1 to capj;          /*This do block puts the death before disease*/
    nps(jj) = ps(1,jj);     /*probs and contributions into the new vectors*/
    newp(jj) = np(1,jj);

end;
                            /*This do block assigns the upper triangular*/
do i = 1 to capi;           /*portion of the possible cells.            */
   do j= i to capj;
       s = i*capj - ((i-1)*i/2.0) +j;
      nps(s) = ps(i+1,j);
      newp(s) = np(i+1,j);

   end;
end;
keep  nps1-nps&total  newp1-newp&total ;
run;

/**Since the vector of probabilities must add to 1, the vector of***
***probabilities used for the information matrix is of length    ***
***total - 1.                                                    ***
*********************************************************************/
data &inform;
set newpseud end=last;
array nps{&total};
array newp{&total};
array info{&infdim,&infdim};
array im{&infdim,&infdim};
do s = 1 to &infdim;
    do t = 1 to &infdim;
      info(s,t) = (nps(s)/newp(s)) - (nps(&total)/newp(&total));
      info(s,t) = info(s,t)*((nps(t)/newp(t)) - (nps(&total)/newp(&total)));
      im(s,t) + info(s,t);
    end;
end;
if last then do;
numcols = &ncols;
numrows = &nrows ;
infdim = &infdim;
end;
```

```
if last;
keep im1-im&infsize newp1-newp&total numrows numcols infdim;
run;
data &inform;
merge &inform rowcol;
run;
%mend;
/*eminfo(pseudata,nrows,ncols,inform) */
/***prepjont.sas***/
/*************************************************************
***This code produces tables of the joint probability     ***
***estimates along with their standard deviations.        ***
***It also fills out the covariance matrices for the      ***
***two groups, in preparation for relative risk and       ***
***lethality calculations.  Finally it does a chi-square  ***
***test on the two sets of joint probabilities and prints ***
***out the p-values and the two joint probability         ***
***estimates.                                             ***
*************************************************************/
%include 'c:\windows\desktop\mortmorb\matrx.sas';
%include 'c:\windows\desktop\mortmorb\estcov.sas';
%include 'c:\windows\desktop\mortmorb\fillout.sas';
%include 'c:\windows\desktop\mortmorb\quaddiff.sas';

%matrx(mm.rinform,rmats);
%matrx(mm.cinform,cmats);  /**Puts information matrices into matrix form.*/
%estcov(rmats,rcov);
%estcov(cmats,ccov);/*creates covariance matrix from information.*/
%pstdtab(rcov,rrow,rcolumn,10,8.6,Exposed Joint Probabilities);
%pstdtab(ccov,crow,ccolumn,10,8.6,Comparison Joint Probabilities);

%fillcov(rcov,rfilld);
%fillcov(ccov,cfilld); /*puts out the complete covariance matrix
                         for all probabilities.*/
%quaddiff(rfilld,cfilld,rrow,rcolumn,joint probs,Exposed,Controls);
                        /*Compares the joint probability distributions*/
/***quaddiff.sas  6/3/98 */
%macro quaddiff(fullcov1,fullcov2,rowfmt,colfmt,measure,lab1,lab2);
/*************************************************************
***This macro computes the quadratic difference between two***
***sets of probability vector estimates using their        ***
***estimated covariance matrices.                          ***
```

```
***Both fullcov1 and fullcov2 are data sets containing the ***
***k rows and columns of a covariance matrix in variables  ***
***named cov1 through covk.  They also contain a variables ***
***named p, row and column.  The variable named p contains ***
***the first k of k+1 probabilities.  The variables row and***
***column conain the row and column number of the original ***
***matrix of rectangles for the probability in p.  The row ***
***and column values should be the same for both fullcov1 and***
***fullcov2, since both data set refer to the same set of  ***
***rectangles.  Letting im1 and p1, im2 and p2 be the      ***
***matrix and probability vector for fullcov1 and 2 resp.  ***
***The output will be the quadratic form of p1-p2 with the ***
***sum of the inverses of im1 and im2.                     ***
***The value will be in a data set called quadiff.        ***
***Measure is the parameter being testd, e.g. lethality,  ***
***lab1 and lab2 that identify the groups.                 ***
************************************************************/
data _null_;
set &fullcov1 nobs=n;
dim = n;
call symput('dim',compress(dim));
stop;
run;

data _cov1_;
set &fullcov1;
keep cov1-cov&dim;
run;

data _p1_;
set &fullcov1;
keep p;
run;

data _rc1_;
set &fullcov1;
keep row column;
run;

data _rc2_;
set &fullcov2;
row2 = row;
```

```
column2 = column;
keep row2 column2;
run;

/***Testing to be sure row and column numbers are the same in***
****both data sets.                                            **/

data rowscols;
merge _rc1_ _rc2_ end=last;
rowdiff = row - row2;
coldiff = column - column2;
sumrowd + rowdiff;
sumcold + coldiff;
if last then do;
     if sumrowd ne 0 then do;
          put '   WARNING ROW NUMBERS ARE NOT THE SAME.';
     end;
     if sumcold ne 0 then do;
          put '   WARNING COLUMN NUMBERS ARE NOT THE SAME.';
     end;
end;
run;

/*****************************************************************/
data _cov2_;
set &fullcov2;
keep cov1-cov&dim;
run;

data _p2_;
set &fullcov2;
keep p;
run;

proc iml;
/****converting fullcov1 to matrix and vector form****/
   use _cov1_;
    read all var _num_ into cov1;
   use _p1_;
    read all var {p}  into p1;
 /*print inf1 p1;*/
/****converting fullcov2 to matrix and vector form****/
```

```
use _cov2_;
    read all var _num_ into cov2;
use _p2_;
    read all var {p}                    into p2;
 /* print inf2 p2;*/

/*computing quadratic difference*/
    cov12 = cov1 + cov2;
    eig = eigval(cov12);  /*eig is the vector of Eigenvalues*/
    /*print eig;*/
   create eig from eig[colname = 'eig'];
    append from eig;     /*eigen values now in data set named eig.*/
    invcov12 = ginv(cov12);

    quadiff = (p1-p2)`*invcov12*(p1-p2);

create quadiff from quadiff;
  append from quadiff;
 /* print quadiff;*/
quit;
/*******Puts vector of estimates in one dataset****/

data _p1_;
set _p1_ ;
p1 = p;
drop p;
run;

data _p2_;
set _p2_ ;
p2 = p;
drop p;
run;

data bothprob;
merge _rc1_ _p1_ _p2_;
run;
/************************/
data _null_;
set quadiff;
call symput ('quadiff',compress(col1));
stop;
```

```
run;

/*****The next data step computes the p-value for the quadratic ***
******difference for the null hypothesis of no difference betwee***
******the two joint probabilities.                                **/

/**first count number of nonzero eigen values to get degrees of freedom.*/
data eignon0;
set eig;
if eig > .000000000000000001;   /**note this is how we define zero**/
run;

data _null_;
set eignon0 nobs = ordr;
call symput('ordr',compress(ordr));
stop;
run;

data _null_;
pvalue = 1.0 - probchi(&quadiff,&ordr);
call symput('pvalue',compress(pvalue));
stop;
run;

proc print data = bothprob noobs;
var row column p1 p2;
title1 "The quadratic difference is &quadiff.";
title2 "It has &ordr degrees of freedom and";
title3 "p-value = &pvalue";
title4 "P1 is &measure from &lab1";
title5 "P2 is &measure from &lab2";
format row &rowfmt.. column &colfmt..;
run;
%mend;

/****example***/
/**Use in the following way.***/
/**%include 'c:\windows\desktop\mortmorb\matrx.sas';
   %include 'c:\windows\desktop\mortmorb\fillout.sas';
   %include 'c:\windows\desktop\mortmorb\estcov.sas';

     %matrx(mm.rinform,rmats); In matrx.sas
```

```
     %matrx(mm.cinform,cmats); These two calls put the information
                               matrices for the k-1 estimates and the
                               k-1 estimates into matrix and vector
                               form.

     %estcov(rmats,rcov); In estcov.sas.
     %estcov(cmats,ccov); inverts the info matrix and keeps the k-1
                          probability estimates.

     %fillcov(rcov,rfilld); In fillout.sas
     %fillcov(ccov,cfilld); Gives the complete covariance matix and
                            for the K probabilities and give the K
                            probability estimates in the correct format
                            for use with quaddiff.
%quaddiff(rfilld,cfilld,rrwo,rcolumn,joint probs,Exposed,Controls);*/
%macro ratiostd(par1,par2,sig1,sig2,cov,outfile);
/************************************************************
***This macro computes the asymtotic standard deviation of***
***the ratio par1/par2 as well as the ratio itself.       ***
***Par1, par2 are the values of the estimators and        ***
***sig1 and sig2 are the variances of par1 and par2,      ***
***respectively.  Cov is the covariance of par1 and par2. ***
***The computation assumes that the vector (par1,par2) is ***
***approximately normal (in the aymtotic sense) with      ***
***covariance matrix given by sig1,sig2 cov.  The variance***
***formula is derived from the delta rule.                ***
***Outfile contains one observation with variable ratio   ***
***and stdev, which are par1/par2 and the standard        ***
***deviation of the ratio.  Macro variables by the same   ***
***name also contian these value.                         ***
************************************************************/
%global ratio stdev;
%local outfile;
data &outfile;
p1 = &par1;
p2 = &par2;
s1 = &sig1;
s2 = &sig2;
c  = &cov;
output;
stop;
run;
```

```
data &outfile;
set &outfile;
ratio = p1/p2;
stdev = s1*s1 - 2*c*ratio + s2*s2*ratio*ratio;
stdev = stdev/(p2*p2);
stdev =sqrt(stdev);
call symput('ratio',compress(ratio));
call symput('stdev',compress(stdev));
keep ratio stdev;
run;
/*
%put The ratio is &ratio;
%put The standard deviation is &stdev; */

%mend;

/* ratiostd(par1,par2,sig1,sig2,cov,outfile);*/
/***realfreq.sas   3/30/98***/
%macro  realfreq(infile,outfile,disfmt,dethfmt,libdrive);
/*********************************************************
***This macro takes an file named in infile that       ***
***contains variables obst, obsd, idr, idL, itr, itL.  ***
***obst and obsd are the observed times for time to    ***
***disease and time to death, respectively.            ***
***Idr and idL are indicators that indicate whether    ***
***time to death is censored on the right and/or left, ***
***respectively.  An indicator equal to 1 indicates    ***
***censored.  Itr and itL are the censoring indicators ***
***for time to disease. The outfile will contain       ***
***proc freq output for the cells with rows for disease***
***and columns for the death variable.                 ***
***The proc freq output will be for the uncensored     ***
***disease and death pairs, including those where      ***
***death is known and disease is known not to have     ***
***occurred during the lifetime, categorized in        ***
***rectangles defined by formats disfmt for the rows   ***
***and dethfmt for the cols.                           ***
*********************************************************/
libname library "&libdrive";
/****The next data step keeps only the uncensored pairs and ***
*****those pairs with uncensored death time and disease time***
```

```
*****known to have occurred after death.                    **/

data actual;
set &infile;
sumd = idr + idl;
sum =idr + idl + itr +itl;
if sum=0 or                    /*disease and death times known*/
     (obst = . and sumd = 0); /*death known to have occurred
                                  before disease and death time known*/
keep obst obsd;
run;

proc freq data = actual /*noprint*/;
title 'Uncensored disease by death counts';
format obst &disfmt.. obsd &dethfmt..;
table obst*obsd/missing sparse out= &outfile;
run;

%mend;
/***This program computes the relative risk ratio for all rectangles in ***
****the matrix of time-to-onset by time-to-death matrix.  One needs to  ***
****run fixboth.sas to get rinform and cinform, the information matrices***
****and probability estimates along with row and column indices for     ***
****Exposed and controls.  The cov matrices are obtained by the     ***
****following sequence.
           %matrx(mm.rinform,rmats);
           %matrx(mm.cinform,cmats);
           %estcov(rmats,rcov);
           %estcov(cmats,ccov);
           %fillcov(rcov,rfillcov);
           %fillcov(ccov,cfillcov);
****The data sets &numcov and &dencov contain the complete covariance***
****matrices,probability estimates and row column indicators in        ***
****variables cov1-covK,p, and row column in K observations.           ***
*************************************************************************/
/****This macro requires that the row and column indices have formats  ***
*****rrow and rcolumn for the original rectangle row and column        ***
*****definitions.  These are outputs from the fixboth macro and should ***
*****be the same as formats crow and ccolumn.                          ***
*************************************************************************/
%macro relrisk(numcov,dencov,title);
/***The following code gets the dimension of the pvector for the two***
```

```
****groups and tests to see if they are equal.                          ***
*************************************************************************/
data _null_;
set &numcov nobs=n;
call symput('rdim',compress(n));
stop;
run;

data _null_;
set &dencov nobs=n;
if n = &rdim then do;
     end = 0;
     call symput('end',compress(end));
     stop;
end;
else do;
     end = 1;
     call symput('end',compress(end));
     stop;
end;
run;

%if &end = 1 %then %do;
  %put  The dimensions of the two probability vectors are unequal.;
  %put  We cannot compute the requested table.;
%end;

%else %do;

data rcovp;
set &numcov;
array cov{&rdim};
array rcov{&rdim};
%do i = 1 %to &rdim;
      rcov(&i) = cov(&i);
%end;
      rp = p;
      rrow = row;
      rcolumn = column;
keep rp rrow rcolumn rcov1-rcov&rdim;
run;
```

```
data ccovp;
set &dencov;
array cov{&rdim};
array ccov{&rdim};
%do i = 1 %to &rdim;
      ccov(&i) = cov(&i);
%end;
      cp = p;
      crow = row;
      ccolumn = column;
keep cp crow ccolumn ccov1-ccov&rdim;
run;


data bothcovp;
merge rcovp ccovp;
run;

/***testing to be sure the row and column indices match.***/

data _null_;
set bothcovp ;
if rrow ne crow or rcolumn ne ccolumn then do;
      end = 1;
      call symput('end',compress(end));
      stop;
end;
run;

%if &end = 1 %then %do;
   %put The row and/or column indices for the two vectors;
   %put Do not match.;
   %put We must stop here.;
%end;

%else %do;

data _ratstd_;
set bothcovp;
array rcov{&rdim};
array ccov{&rdim};
relrisk = rp/cp;
```

```
sig1 = rcov(_n_);
sig2 = ccov(_n_);
var = (sig1 + sig2*relrisk*relrisk)/(relrisk*relrisk);  /*using delta rule*/
stdev = sqrt(var);
row = rrow;
column = rcolumn;
keep relrisk stdev row column;
run;

/***In here we should use the table macro**/

%include 'c:\windows\desktop\mortmorb\tabmac.sas';
%tableit(_ratstd_,row,column,
      rrow,rcolumn,10,8.6,relrisk stdev,time-to-onset,
       time-to-death,&title);
%end;
%end;
%mend;
/*example*/
/*
%include 'c:\windows\desktop\mortmorb\matrx.sas';
%include 'c:\windows\desktop\mortmorb\estcov.sas';
%include 'c:\windows\desktop\mortmorb\fillout.sas';

%matrx(mm.rinform,rmats);
           %matrx(mm.cinform,cmats);
           %estcov(rmats,rcov);
           %estcov(cmats,ccov);
           %fillcov(rcov,rfillcov);
           %fillcov(ccov,cfillcov);
%relrisk(rfillcov,cfillcov,Relative Risk Exposed to Controls); */
/****revzero.sas   3/30/98**/
%macro nextzero(probs,times,numrows,numcols,numtimes,
                                    probpref,newprobs,newtimes);
/*********************************************************
***This macro checks to see if the oldrows by oldcols ***
***matrix  of probabilities in data set probs has any ***
***zero cells in the first row or in the diagonal and ***
***upper diagonal of the lower oldrows -1 by oldcols  ***
***matrix.  Probpref is the prefix for this array of  ***
***probabilities. times contains the array of endpoint***
***for the time intervals with prefix t.              ***
```

```
*** As soon as a zero cell is found the                    ***
***matrix is collapsed by combining the column with        ***
***the zero entry with an adjacent column.  If there       ***
***is a choice of columns to merge with the zero cell      ***
***column the column whose cell with the same row as       ***
***the zero cell with the smallest probability is used***
***to combine with the zero cell.  After collapsing        ***
***the process of finding zero cells and combining         ***
***columns is continued until the resulting matrix has***
***no zero cells in the first row or diagonal or upper***
***diagonal of the matrix below the first row.             ***
***The new probability matrix is in a data set newprobs***
***with prefix newp.  The new number of rows and            ***
***columns are in global macro variables named newrows***
***and newcols.  The dataset newtimes contains the         ***
***new times that form the endpoints for the row and       ***
***column intervals in an array t.  The new number ***
***of endpoints is in the macro variable newtme.           ***
***oldtimes is the number of endpoints used for the        ***
***disease and death intervals.                            ***
***A global macro variable yes is produced.  Yes = 1       ***
***if a zero cell was found and thus collapsing done. ***
***Yes = 0 if no collapsing was done.  If Yes = 0 then***
***the output matrix of probabilities and the output       ***
***array of times is the same as the input.                ***
************************************************************/
data _null_;
   oldtot = &numrows*&numcols;
   oldrows = &numrows;
   oldcols = &numcols;
   oldtimes = &numtimes;
   call symput('oldtot',compress(oldtot));
   call symput('oldrows',compress(oldrows));
   call symput('oldcols',compress(oldcols));
   call symput('oldtimes',compress(oldtimes));
stop;
run;

data _probs_;  /**Putting original rectangle probs into _probs_*/
set &probs;
 array &probpref.{&oldrows,&oldcols}; /*probpre is the prefix for
                                        the cell probs.*/
```

```
 array oldp{&oldrows,&oldcols};
   do i  = 1 to &oldrows;
     do j=1 to &oldcols;
         oldp(i,j) = &probpref.(i,j);    /*Renaming the old probs.*/
     end;
   end;
 keep oldp1-oldp&oldtot;
run;

data _probs_; /**Putting original times into _times_.*/
              /*and creating working data set*/
merge _probs_ &times;
 array t{&oldtimes};
 array oldt{&oldtimes};
    do k = 1 to &oldtimes; /**renaming original times**/
      oldt(k) = t(k);
    end;
drop t1-t&oldtimes;
run;

%global newrows newcols newtot newtime yes;
%let redo = 1;
%let first = 1;   /*This indicates that we are starting
                  **the first pass at checking for zeros.*/

%do %while(&redo=1);  /*continue searcing process if redo is 1.*/
                      /*Redo changes to 0 when no upper diagonal
                      **zero cells are left.**/
/*************************************************************
****The following data step find the first zero cell if any***
****and puts the row and column number of the cell in macro***
****variables row and col.                                  ***
*************************************************************/
data _null_;
set _probs_;
array oldp(&oldrows,&oldcols);
first = &first;
do j= 1 to &oldcols;              /***checking first row**/
    if oldp(1,j) = 0 then do;
          i=1;
          call symput('row',compress(i));
          call symput('col',compress(j));
```

```
        if first then do;
           yes = 1;
           call symput('yes',compress(yes));
           first = 0;
           call symput('first',compress(first));
        end;
           stop;
         end;
      end;
do i=2 to &oldrows;          /*checking lower sumatrix*/
    do j= i-1 to &oldcols;   /*Only the diagonal and upper diagonal.*/
    if oldp(i,j) = 0 then do;
           call symput('row',compress(i));
           call symput('col',compress(j));
        if first then do;
           yes = 1;
           call symput('yes',compress(yes));
           first = 0;
           call symput('first',compress(first));
        end;
           stop;
         end;
       end;
      end;
redo = 0;          /*We only get here if no zeros found in which case
                   ** this is the last search.*/
 if first then do;
           yes = 0;
           call symput('yes',compress(yes));
        end;
call symput('redo',compress(redo));
stop;
run;

/*************************************************************
***We begin combining column col with column col-1 or col+1.***
***Keep in mind that the interval for column col has        ***
***endpoints times(col-1) and times(col+1).  The            ***
***corresponding row if oldrows >= col+1 is row equal col+1.***
***Regardless of which columns are collapsed oldcols will be***
***reduced by one, the times will be reduced by 1.  The     ***
***number of rows will be reduced by 1 if oldrows>col+1.    ***
```

```
*************************************************************/
%if &redo = 1 %then %do;
/*%if %eval(&oldrows) > %eval(&col + 1) %then %do;
       %let newrows = %eval(&oldrows - 1);
     %end;
   %else %do;
       %let newrows = %eval(&oldrows);
     %end;*/
/******The following code determines whether we merge the zero*****
*******column with the right column or with the left column.  *****
*******The macro variable merge is 1 if we merge right and -1 *****
*******if we merge left.                                      *****
*************************************************************/

data _null_;
set _probs_;
array oldp{&oldrows,&oldcols};
oldrows = &oldrows;
oldcols = &oldcols;
row = &row;
col = &col;
if col = 1 or row = col+1 or (1 < col < oldcols  and
                    oldp(row,col-1) > oldp(row,col+1)) then
                            merge = 1;
else merge = -1;
/****Now we determine the number of newrows.**/
if merge = 1 then do;
          if oldrows > col + 1  then newrows = oldrows - 1;
          else                       newrows = oldrows;
end;
else if merge = -1 then do;
          if oldrows >= col + 1 then newrows = oldrows -1;
          else                       newrows = oldrows;
end;
call symput('merge',compress(merge));
call symput('newrows',compress(newrows));
run;


%let newcols = %eval(&oldcols - 1);
%let newtime = %eval(&oldtimes - 1);
```

```
/*****************************************************************
*****************Now the collapsing begins.************************
*****************************************************************/

   data _probs_;
   set _probs_;
   array oldp{&oldrows,&oldcols};
   array newp{&newrows,&newcols}; /*array of newprobs*/
   array oldt{&oldtimes};
   array newt{&newtime};   /*array to contain newtimes*/
   row = &row;      /*row where zero is*/
   col = &col;      /*column where zero is*/
   newrows = &newrows;
   newcols = &newcols;
   merge = &merge;

      if (col=1 or ( col=2 and merge = -1))
      then  do;                          /*merging old cols 1 and 2*/
                        /*If col is one we can only combine
                        **col 1 and 2.  We combine col 1 and 2
                        **also col=2 old prob is smaller in row 1*/

           do k = 1 to &newtime;
                     newt(k) = oldt(k+1); /*all times move up 1*/
                end;
           newp(1,1)= oldp(1,1) + oldp(1,2);
           newp(2,1)= oldp(2,1) + oldp(2,2) +
                     oldp(3,1) + oldp(3,2);
           if newrows >= 3 then do;
                   do i = 3 to &newrows;
                      newp(i,1) = oldp(i+1,1) + oldp(i+1,2);
                   end;
                end;

                  do j = 2 to &newcols;
                      newp(1,j) = oldp(1,j+1);
                      newp(2,j) = oldp(2,j+1) + oldp(3,j+1);
               if newrows >= 3 then do;
                   do i = 3 to &newrows;
                      newp(i,j) = oldp(i+1,j+1);
                   end;
                end;
```

```
            end;

   end;
else if (col=2 and merge = 1)
   then do;
       /*merging old columns 2 and 3.*/

               newt(1) = oldt(1);
            do k = 2 to &newtime;
               newt(k) = oldt(k+1);
            end;
          do i = 1 to 2;
            newp(i,1) = oldp(i,1);
            newp(i,2) = oldp(i,2) + oldp(i,3);
              if &newcols >=3 then do;
                   do j = 3 to &newcols;
                      newp(i,j) = oldp(i,j+1);
                   end;
              end;
           end;
      if &oldrows>= 4 then do;
           newp(3,1) = oldp(3,1) + oldp(4,1);
           newp(3,2) = oldp(3,2) + oldp(3,3) +
                       oldp(4,2) + oldp(4,3);
       if newcols >= 3 then do;
           do j=3 to &newcols;
             newp(3,j) = oldp(3,j+1) + oldp(4,j+1);
            end;
        end;
        end;
      else do;
           newp(3,1) = oldp(3,1);
           newp(3,2) = oldp(3,2) +oldp(3,3);
        if newcols >= 3 then do;
             do j = 3 to &newcols;
                 newp(i,j) = oldp(i,j+1);
             end;
           end;
      end;
       if newrows >=4 then do;
         do i=4 to &newrows;
           newp(i,1) = oldp(i+1,1);
```

```
            newp(i,2) = oldp(i+1,2) + oldp(i+1,3);
          if newcols >= 3 then do;
              do j= 3 to &newcols;
                newp(i,j) = oldp(i+1,j+1);
              end;
           end;
          end;
         end;
        end;

else if (col = &oldcols or
          ( 2<col<&oldcols and merge = -1) )
     then do;
       /*combines*/
                                          /*col-1 and col*/

         do k = 1 to col-2;
            newt(k) = oldt(k);
         end;
          do k = col-1 to &newtime;
                newt(k) = oldt(k+1);
            end;
          do i = 1 to min(&newrows,col-1);

           do j = 1 to col-2;
              newp(i,j) = oldp(i,j);
           end;
          end;
          do i = 1 to min(&newrows,col-1);
              newp(i,col-1) = oldp(i,col-1) + oldp(i,col);
                if col < &oldcols then do;
                     do j = col to &newcols;
                       newp(i,j) = oldp(i,j+1);
                     end;
                 end;
         end;
       if &oldrows = col then do;
             do j = 1 to col-2;
                 newp(col,j) = oldp(col,j);
             end;
                 newp(col,col-1) = oldp(col,col-1) + oldp(col,col);
          if col < &oldcols then do;
```

```
            do j = col to &newcols;
                newp(col,j) = oldp(col,j+1);
            end;
          end;
        end;
        if &oldrows >= col +1  then do;
          do j = 1 to col-2;
              newp(col,j) = oldp(col,j) + oldp(col+1,j);
            end;
           newp(col,col-1) = oldp(col,col-1) + oldp(col,col) +
                                oldp(col+1,col-1) + oldp(col+1,col);
         if col < &oldcols then do;
           do j = col to &newcols;
             newp(col,j) = oldp(col,j+1) + oldp(col+1,j+1);
          end;
         end;
        if newrows >= col + 1 then do;
          do i = col +1 to &newrows;
             do j = 1 to col-2;
              newp(i,j) = oldp(i+1,j);
             end;
          end;

          do i = col+1 to &newrows;
               newp(i,col-1) = oldp(i+1,col-1) + oldp(i+1,col);
          end;
          do i = col+1 to &newrows;
           if col < &oldcols then do;
             do j = col to &newcols;
               newp(i,j) = oldp(i+1,j+1);
             end;
           end;
          end;
        end;
       end;
      end;
   else if (2<col<&oldcols and merge = 1)
      then do;   /*combine columns col and col+1*/

        do k = 1 to col-1;
          newt(k) = oldt(k); /*times the same up to t(col-1)*/
        end;
```

```
    do k = col to &newtime;      /*lose the old t(col)*/
          newt(k) = oldt(k+1);
      end;
    do i = 1 to min(&newrows,col);
         newp(i,col) = oldp(i,col) + oldp(i,col+1);
     do j = 1 to col-1;
        newp(i,j) = oldp(i,j);

     end;
    end;
   do i = 1 to min(&newrows,col);
    do j = col+1 to &newcols;
        newp(i,j) = oldp(i,j+1);
    end;
   end;
if newrows >= col+1 then do;
    if newrows = col+1 then do;
    do j = 1 to col-1;
                                        /*if col+1 is last new row*/
        newp(col+1,j) = oldp(col+1,j) ;/*then new prob = old for*/
                                        /*columns up to col in row*/
    end;                                /*col + 1.*/

        newp(col+1,col) = oldp(col+1,col) + oldp(col+1,col+1);
    do j = col+1 to &newcols;
        newp(col+1,j) = oldp(col+1,j+1) ;
    end;
     end;
 else do;
      do j = 1 to col-1;
        newp(col+1,j) = oldp(col+1,j) + oldp(col+2,j);
      end;
       newp(col+1,col) = oldp(col+1,col) + oldp(col+1,col+1) +
                         oldp(col+2,col) + oldp(col+2,col+1);
      do j = col+1 to &newcols;
       newp(col+1,j) = oldp(col+1,j+1) + oldp(col+2,j+1);
   end;
  end;
 end;
  if newrows > col + 2 then do;
   do i = col +2 to &newrows;
       do j = 1 to col-1;
```

```
                    newp(i,j) = oldp(i+1,j);
                   end;
               end;

               do i = col+2 to &newrows;
                    newp(i,col) = oldp(i+1,col) + oldp(i+1,col+1);
               end;
               do i = col+2 to &newrows;
                  do j = col+1 to &newcols;
                    newp(i,j) = oldp(i+1,j+1);
                  end;
               end;
               end;
            end;

 newtot = &newrows*&newcols;
 call symput('newtot',compress(newtot));
 run;         /**Need to check all %ends and ends**/

data _probs_;
     set _probs_;
     keep newp1-newp&newtot  newt1-newt&newtime ;
     run;

data _probs_;
set _probs_;   /***putting current probs and times in old array
               **for the next do while loop.**/
array newp{&newtot};
array oldp{&newtot};
array newt{&newtime};
array oldt{&newtime};
do i = 1 to &newtot;
     oldp(i) = newp(i);
 end;
do k = 1 to &newtime;
     oldt(k) = newt(k);
 end;
   oldtot = &newtot;
   oldrows = &newrows;
   oldcols = &newcols;
   oldtimes = &newtime;
   call symput('oldtot',compress(oldtot));
```

```
   call symput('oldrows',compress(oldrows));
   call symput('oldcols',compress(oldcols));
   call symput('oldtimes',compress(oldtimes));
keep oldp1-oldp&newtot oldt1-oldt&newtime;
run;
 %end;  /**This ends the do if redo=1 group**/
 %end;  /**This ends the do while redo=1 group**/

/*****************************************/
/**attempt to fix numcols problem from SA**/
%if &yes = 0 %then %do;
   data _null_;
   newrows = &oldrows;
   newcols = &oldcols;
   call symput('newrows',compress(newrows));
   call symput('newcols',compress(newcols));
   stop;
   run;
%end;
/*****************************************/

     data &newprobs;
     set _probs_;
     array oldp{&oldtot};
     array newp{&oldtot};
     newrows = &newrows;
     newcols = &newcols;
     do i = 1 to &oldtot;
        newp(i) = oldp(i);
     end;
     keep newp1-newp&oldtot newrows newcols;
     run;

     data &newtimes;
     set _probs_;
     array oldt{&oldtimes};
     array t{&oldtimes};
    newrows = &newrows;
     newcols = &newcols;
     do k = 1 to &oldtimes;
        t(k) = oldt(k);
     end;
```

```
     keep t1-t&oldtimes newrows newcols;
     run;
%mend;

/* nextzero(probs,times,numrows,numcols,numtimes,
                                    probpref,newprobs,newtimes);*/
/*%nextzero(fnlprobs,times,&numrows,&numcols,&numtimes,
                                     newp,newprobs,newtimes); */

data mm.exposed mm.control;
set mm.cancer;
if id = 'E' then output mm.exposed;
else if id = 'C' then output mm.control;
run;
%macro submtrx(infile,Bi,Bj,Ei,Ej,outfile);
/************************************************
***This macro produces the covariance matrix for***
***the estimates of the subvector of joint      ***
***probability estimates for a matrix of        ***
***rectangles. It also pulls out the            ***
***corresponding probabilities and their        ***
***their original row and column idices.        ***
***Infile contains in variables named cov1-covK ***
***the covariance matrix for the vector of joint***
***probability estimates whose values are in    ***
***a variable called p.  These are the first K  ***
***probabilities of K+1 probabilities.          ***
***infile must also contain variables Row and Column ***
***which are the rectangle coordinates for the  ***
***estimates in p.                              ***
***Bi and Bj are the row and column coordinates ***
***for the first rectangles whose probability is***
***to be included.  Ei and Ej are the row and   ***
***column coordinates for the last probability  ***
***to be included.  The set of probabilities    ***
***defined by the rectacle of coordinates from  ***
***(Bi,Bj) to (Ei,Ej) will be pulled from the   ***
***p variable in infile.  The corresponding     ***
***submatrix will be pulled from the columns    ***
***cov1-covK.  If the set of coordinates        ***
***the (K+1)st prbability then the respective   ***
***variances and covariances will be computed.  ***
```

```
***output will be a sas data set containing     ***
***the submatrix in the variables names subcov1 ***
***through subcovS, where S is the number of    ***
***probablities in the subvector.  A variable   ***
***named p will contain the subvector of        ***
***probabilities.  Row and Column will be       ***
***variables containing the coordinates of the  ***
***probabilities in the original matrix of      ***
***joint probabilities.                         ***
************************************************/
Data _input_;
set &infile nobs = n end=done;
if _n_ = 1 then do;
    call symput('origdim',compress(n)); /*origdim is the dimension of the
                                  original covariance matrix.  There
                                 are thus origdim+1 joint probabilities.*/
end;
if ( &Bi<= Row <= &Ei) and (&Bj <= Column <= &Ej) then do;
    _keep_ = _n_;
end;
else _keep_ = 0;
run;

data _keeprs_;
set _input_;
if _keep_ ne 0;
keep _keep_;
run;

data _null_;
set _keeprs_ nobs = n;
call symput('count',compress(n));
stop;
run;

data _keeprs_;
set _keeprs_ end = last;
array subp{&count};
retain subp1-subp&count;
subp(_n_) = _keep_;
if last;
keep subp1-subp&count;
```

```
run;

data _null_;
set _keeprs_;
array subp{&count};
%do i = 1 %to &count;
    call symput("keep&i",compress(subp(&i)));
%end;
stop;
run;

Data &outfile;
set _input_;
if _keep_ ne 0;
keep row column p;
%do i = 1 %to &count;
    keep cov&&keep&i;
%end;
run;
%mend;

/*submtrx(infile,Bi,Bj,Ei,Ej,outfile);*/
/*options notes symbolgen;
%submtrx(rcov,2,2,3,3,out); */
%macro sumstd(covprc,prefix,outfile);
/*******************************************************
***This macro computes the sum and standard deviation***
***of the sum of probability estimates.               ***
***covprc is a SAS data set with columns prefix1      ***
***through prefixK containing the covariance matrix   ***
***of the probabilities in the variable p, where K    ***
***is the number of observations (and probabilities).***
***covprc also contains variables row and column      ***
***Outfile will be a SAS data set two variables whose***
***names are sum and stdev and one observation.       ***
***Sum and stdev will                                 ***
***be the names of global macro variables containing  ***
***the sum of the probabilities and the standard      ***
***deviation of the sum.                              ***
*******************************************************/
%global sum stdev;
proc contents data = &covprc out=_conts_ noprint;
```

```
run;

%let preflen = %length(&prefix);
data _conts_;
set _conts_;
retain _pref_;
_pref_ =  upcase(substr(name,1,&preflen));
if _pref_ = upcase("&prefix");
keep name;
run;

data _null_;
set _conts_ nobs=n;
call symput('n',compress(n));
stop;
run;

data _conts_;
set _conts_ end=last;
array cov{&n} $;
retain cov1-cov&n;
cov(_n_) = name;
if last;
run;

data _null_;
set _conts_;
array cov{&n} $;
%do i = 1 %to &n;
   call symput("cov&i",compress(cov(&i)));
%end;
stop;
run;        /*names of the covariance columns are in macros cov1-covn*/

data &outfile;
set &covprc end=last;
rowtot = 0;
%do i = 1 %to &n;
    rowtot= rowtot + &&cov&i;
%end;
     variance + rowtot;
     sum + p;
```

```
if last then do;
     stdev = sqrt(variance);
     call symput('sum',compress(sum));
     call symput('stdev',compress(stdev));
end;
if last;
keep sum stdev;
run;
/*
%put   The sum of the probalities is &sum;
%put   The standard deviation is &stdev;  */
%mend;

/*sumstd(covprc,prefix,outfile)*/
data test;
input dan1-dan5 p;
cards;
      1 2 3 4 5 .05
      2 3 4 5 1 .1
      3 4 5 1 2 .2
      4 5 1 2 3 .25
      5 1 2 3 4 .3
;
run;

%sumstd(test,dan,out);
%macro tableit(infile,class1,class2,fmt1,fmt2,collen,cellfmt,var,
               label1,label2,title);
/************************************************************
***collen is the legth of the column headings.
***cellfmt is the format for the cell values.
***var is the list variables to be included in the cells.
***eg if prob and std are to be in the cells then var=prob cell.
***Label1 and label two are the names of the two class variables.
***For example label1 might be time-to-onset and label2 might
***be time-to-death.
***Title is the title of the table.*/
options ls=78 missing='0' nodate;

%let rts = %eval(&collen + &collen + 4);

proc tabulate data = &infile format=&cellfmt ;
```

```
format &class1 &fmt1.. &class2 &fmt2..;
class &class1 &class2;
var &var;
table &class1*(&var), sum*&class2/rts= &rts ;
keylabel sum = '  ';
label &class1="&label1" &class2="&label2";
title "&title";
run;

options missing = '.';
%mend;


/*tableit(infile,class1,class2,fmt1,fmt2,collen,cellfmt,var,
              label1,label2,title);*/
/**Exmaple--see tabling.sas for max and test2
%tableit(test2,ni,nj,interv,interv,&max,&max..4,prob std,time-to-onset,
      time-to-death,Joint probability table); */
/**newtime5.sas   3/30/98*/

%macro ddformat(infile,numt,numdis,numdeth,pref);
/*************************************************************
***This macro takes an infile that contains ordered times t1 ***
***through t&numt and creates formats for disease and death  ***
***times called &pref.dis and &pref.deth, resp.  Disease has ***
***numdis rows of which the first row is disease missing and ***
***the other intervals are defined by the ts.                ***
***Death has no missing values and numdeth intervals defined ***
***by the ts.                                                ***
*************************************************************/
/**The following data step puts the times from infile to be used***
***as endpoints into macro variables ti.*/
data null;
set &infile;
%do i = 1 %to &numt;
call symput("t&i",trim(left(t&i)));

%end;
stop;
run;

/**&pref.deth is the format for the death times.
```

```
                                        No missing values allowed.**/
proc format library = library;
    value &pref.deth        0 -< &t1 =        "[0,&t1)"
%do i = 2 %to %eval(&numdeth - 1) ;
          %let ii = %eval(&i - 1);
                      &&t&ii -< &&t&i = "[&&t&ii,&&t&i)"
%end;
          %let jj = %eval(&numdeth - 1);
                      &&t&jj - high = "[&&t&jj,+)" ;
run;

/**&pref.dis is the format for the disease times plus a missing category.  **/

proc format library = library;
    value &pref.dis                . ="death w/o disease"
                      0 - <&t1 =        "[0,&t1)"
%do i = 2 %to %eval(&numdis - 2);
          %let ii = %eval(&i - 1);
             &&t&ii - <&&t&i = "[&&t&ii,&&t&i)"
%end;
           %let jj = %eval(&numdis - 2);
                      &&t&jj - high = "[&&t&jj,+)";
run;
/****The following code assigns the intervals defining the rectangles***
*****to row and column numbers.  These formats can be used later when***
*****tabling functions of the joint probability distribution with    ***
*****only knowledge of the row and column numbers.                   **/
proc format library = library;
    value &pref.column        1 =        "[0,&t1)"
%do i = 2 %to %eval(&numdeth - 1) ;
          %let ii = %eval(&i - 1);
                      &i = "[&&t&ii,&&t&i)"
%end;
          %let jj = %eval(&numdeth - 1);
                      &numdeth = "[&&t&jj,+)" ;
run;

/**&pref.dis is the format for the disease times plus a missing category.  **/

proc format library = library;
    value &pref.row                1 ="death w/o disease"
                                   2 =        "[0,&t1)"
```

```
%do i = 2 %to %eval(&numdis - 2);
          %let ii = %eval(&i - 1);
          %let iii = %eval(&i + 1);
                &iii= "[&&t&ii,&&t&i)"
%end;
           %let jj = %eval(&numdis - 2);
                     &numdis = "[&&t&jj,+)";
run;
%mend;
%macro  eminvals(infile,outfile,timesout,intobs,libdrive);
/*********************************************************
***This macro takes an file named in infile that       ***
***contains variables obst, obsd, idr, idL, itr, itL.  ***
***obst and obsd are the observed times for time to    ***
***disease and time to death, respectively.            ***
***Idr and idL are indicators that indicate whether    ***
***time to death is censored on the right and/or left, ***
***respectively.  An indicator equal to 1 indicates    ***
***censored.  Itr and itL are the censoring indicators ***
***for time to disease. The outfile will contain       ***
***proc freq output for the cells with rows for disease***
***and columns for the death variable.                 ***
***Intobs is the number of uncensored observations     ***
***desired between the points on the disease and death ***
***interval. Timesout is a file containing the times   ***
***used to create the proc freq table and format.      ***
***Disease times and deathtimes are chosen separately  ***
***with intomb uncensored times between them.  Then    ***
***they are put into one set of sorted points.  The    ***
***resulting contingency table will be upper right     ***
***triangular.                                         ***
***NOTE:  If this macro is used in a simulation then   ***
***remove the comments around no print in the proc freq***
***statement at the end of the macro.                  ***
*********************************************************/
libname library "&libdrive";
/****The next data step keeps only the uncensored pairs and ***
*****those pairs with uncensored death time and disease time***
*****known to have occurred after death.                    **/

data actual;
set &infile;
```

```
sumd = idr + idl;
sum =idr + idl + itr +itl;
if sum=0 or                    /*disease and death times known*/
     (obst = . and sumd = 0); /*death known to have occurred
                                  before disease and death time known*/
keep obst obsd;
run;

/***This data step keeps only the uncensored disease times.***/
data distimes;
set actual;
if obst ne .;
keep obst;
run;

/**Uncensored disease times are sorted next.**/

proc sort data=distimes;
by obst;
run;

/***The user indicates he/she want intobs in each interval.  ***
****The following data step pulls the intobsth. time from the***
****sorted list.                                              **/

data distimes;
set distimes ;
if not(mod(_n_,&intobs)) ;
time = obst;
keep time ;
run;

/***The following data step sort and data step repeat the process ***
****just completed for disease times.***/
data dethtime;
set actual;
keep obsd;
run;

proc sort data=dethtime;
by obsd;
run;
```

```
data dethtime;
set dethtime ;
if not(mod(_n_,&intobs)) ; /*We also choose the largest disease
                                    and death times--not.*/
time = obsd;
keep time ;
run;

/***The next data step combines the disease and deathtime endpoints.***
****This is similar to what Louis et. al. number III suggest for    ***
****choosing intervals.                                             **/

data combined;
set dethtime distimes;
run;

proc sort data = combined;
by time;
run;

/****repeated values of combined disease and death uncensored times***
***** are removed next.                                            **/

data combined;
set combined;
if _n_ = 1 then b = time;
else do;
   if time = b then delete;
   else b = time;
end;
retain b;
drop b ;
run;

/***Number of distinct times is put into macro variable numtimes.***/

data null;
set combined end = last nobs = n;
call symput('numtimes',left(trim(n)));
stop;
run;
```

```
/***Output data set of times is created next.  timesout has one ***
****observation which is an array of length numtimes having     ***
****prefix t.                                                  **/

data &timesout;
set combined end = last;
array t{&numtimes};
retain t1-t&numtimes;
t(_n_) = time;
if last;
keep t1-t&numtimes;
run;

/*************************************************************
***We now use the points ti in the outfile from the        ***
***macro eminvals to format T and D, where T is time to    ***
***disease and D is time to death.    We use this format   ***
***in a PROC FREQ to begin reducing the intervals to a     ***
***lattice that has a reasonable number of uncensored      ***
***points in each rectangle.                               ***
***Clasfile will be an outfile.  It is the original datafile***
***with variables row and column indicating the row and    ***
***column membership for obst and obsd   The row is the    ***
***disease category and column is the death category.      ***
***Disease row= 1 means missing.  Rowl, rowr, columnl and  ***
***columnr are the similar variables for obstl, obstr,     ***
***obsdl and obsdr.                                        ***
*************************************************************/
%let numdis = %eval(&numtimes +2);
%let numdeth = %eval(&numtimes + 1);
%ddformat(&timesout,&numtimes,&numdis,&numdeth,&pref);

/**put comments around noprint for simulation runs.**/

proc freq data = actual /*noprint*/;
title 'Uncensored disease by death counts';
format obst &pref.dis. obsd &pref.deth.;
table obst*obsd/missing sparse out= &outfile;
run;
%mend;
```

```
/* eminvals(infile,outfile,times,intobs) */
/*%eminvals(mortmorb,times,20);*/
%macro totrlrsk(numcovp,dencovp,outfile,title);
/*****************************************************
***This macro takes two matrices of estimated joint  ***
***probabilities and produces the relative risk of   ***
***dying with disease for numcovp vs dencovp groups. ***
***numcovp is a SAS data set for the numerator of the***
***relative risk.  dencovp is a SAS data set for the ***
***denominator of the relative risks.  Both SAS data ***
***sets must have the same number or rows and columns***
***corresponding to the same set of rectangles in a  ***
***two dimensional matrix.  The columns for the data ***
***sets must be variables named cov1-covK,p, row and ***
***column.  K is the number of joint non-zero joint  ***
***probabilities for the rectangles and also the     ***
***number of rows.  Cov1-covK are the columns of the ***
***covariance matrix for the estimates of the K joint***
***probabilities contained in variable p.  Row and   ***
***column are the row and column indices, indicating ***
***the coordinates of the rectangle for p is the     ***
***estimate of the joint probability.  The row and   ***
***column values for both data sets must be the same ***
***and in the same order.                            ***
***A SAS data set given the name in the argument     ***
***outfile will contain one observation with         ***
***variables relrisk, stdev and column.              ***
***Title will be the title of the printed output.    ***
*****************************************************/
/**WARNING!!!   WARNING!!!    WARNING!!!   WARNING!!!***
*** Be sure that the follwoing include statements    ***
***have the correct addresses.                       ***
***WARNING!!!   WARNING!!!    WARNING!!!   WARNING!!!**/
      %include 'c:\windows\desktop\mortmorb\submtrx.sas';
      %include 'c:\windows\desktop\mortmorb\sumstd.sas';
      %include 'c:\windows\desktop\mortmorb\ratiostd.sas';
/***The following code gets the dimension of the pvector for the two***
****groups and tests to see if they are equal.                      ***
*************************************************************************/
data _null_;
set &numcovp nobs=n end = last;
call symput('rdim',compress(n));
```

```
if last then do;
   call symput('colnum',compress(column)); /**number of columns put into
                                            macro variable colnum**/
   call symput('rownum',compress(row)); /**number rows in rownum**/
end;
run;

data _null_;
set &dencovp nobs=n;
if n = &rdim then do;
     end = 0;
     call symput('end',compress(end));
     stop;
end;
else do;
     end = 1;
     call symput('end',compress(end));
     stop;
end;
run;

%if &end = 1 %then %do;
  %put  The dimensions of the two probability vectors are unequal.;
  %put  We cannot compute the requested table.;
%end;
%else %do;
data covnum;   /****r is the prefix for the numerator.**/
set &numcovp;
array cov{&rdim};
array rcov{&rdim};
%do i = 1 %to &rdim;
      rcov(&i) = cov(&i);
%end;

      rrow = row;
      rcolumn = column;
run;
data covden;        /***c is the prefix for the denominator.**/
set &dencovp;
array cov{&rdim};
array ccov{&rdim};
%do i = 1 %to &rdim;
```

```
      ccov(&i) = cov(&i);
%end;

      crow = row;
      ccolumn = column;
run;

data bothcovp;
merge covnum covden;
run;

/***testing to be sure the row and column indices match.***/

data _null_;
set bothcovp ;
if rrow ne crow or rcolumn ne ccolumn then do;
      end = 1;
      call symput('end',compress(end));
      stop;
end;
run;

%if &end = 1 %then %do;
   %put The row and/or column indices for the two vectors;
   %put Do not match.;
   %put We must stop here.;
%end;

%else %do;
      data covnum;
      set covnum;
      row=rrow;
      column = rcolumn;
      drop rrow rcolumn rcov1-rcov&rdim ;
      run;

      data covden;
      set covden;
      row=crow;
      column = ccolumn;
      drop crow ccolumn ccov1-ccov&rdim;
      run;
```

```
      %submtrx(covnum,2,1,&rownum,&colnum,numjcov);
      %submtrx(covden,2,1,&rownum,&colnum,denjcov);
      %sumstd(numjcov,cov,numout);
      %sumstd(denjcov,cov,denout);
/**The files numout and denout contain one observation each***
***with variables sum and stdev, which are the sum of the  ***
***probabilities in the column and the standard deviation  ***
***of the sum, respectively.  The next code puts these     ***
***quantities into macro variables with the appropriate    ***
***names.                                                  ***
************************************************************/

      data _null_;
      set numout;
      sig = stdev*stdev;
      call symput('numsig',compress(sig));
      call symput('numsum',compress(sum));
      stop;
      run;

      data _null_;
      set denout;
      sig = stdev*stdev;
      call symput('densig',compress(sig));
      call symput('densum',compress(sum));
      stop;
      run;

/***The next call to ratiostd computes the relative risk for row j***
****along with its standard deviation.                          ***
************************************************************************/
      %ratiostd(&numsum,&densum,&numsig,&densig,0,ratout);
/***Now the relative risk and standard deviation will be put into***
****the output data set.                                        ***
************************************************************************/

      data &outfile;
      label relrisk = 'Relative Risk' stdev = 'Standard Deviation';
      set ratout;
      relrisk = ratio;

      keep relrisk stdev;
```

```
      run;

      proc print data = &outfile noobs label;
      title "&title";
      var relrisk stdev;
      run;
  %end;
%end;
%mend;

/*example*/
/*totrlrsk(numcovp,dencovp,outfile,title) */
libname library 'c:\windows\desktop\mortmorb'; /*where formats are stored
/*options nosymbolgen nonotes; */
/*%totrlrsk(rfilld,cfilld,final, Total Relative Risk of Dying with disease);*/
libname library 'c:\windows\desktop\mortmorb';
proc format library = library;
      value dis  . = 'death wo disease'
             0- 15 = '[0,15)'
            15- 20 = '[15,20)'
            20- 25 = '[20,25)'
            25 -30 = '[25,30)'
            30- 35 = '[30,35)'
            35- high = '[35,+)';
run;
proc format library = library;
  value deth  0- 15 = '[0,15)'
            15- 20 = '[15,20)'
            20- 25 = '[20,25)'
            25 -30 = '[25,30)'
            30- 35 = '[30,35)'
            35- high = '[35,+)';
run;

data times;
input t1-t5;
cards;
   15 20 25 30 35
;
run;
```